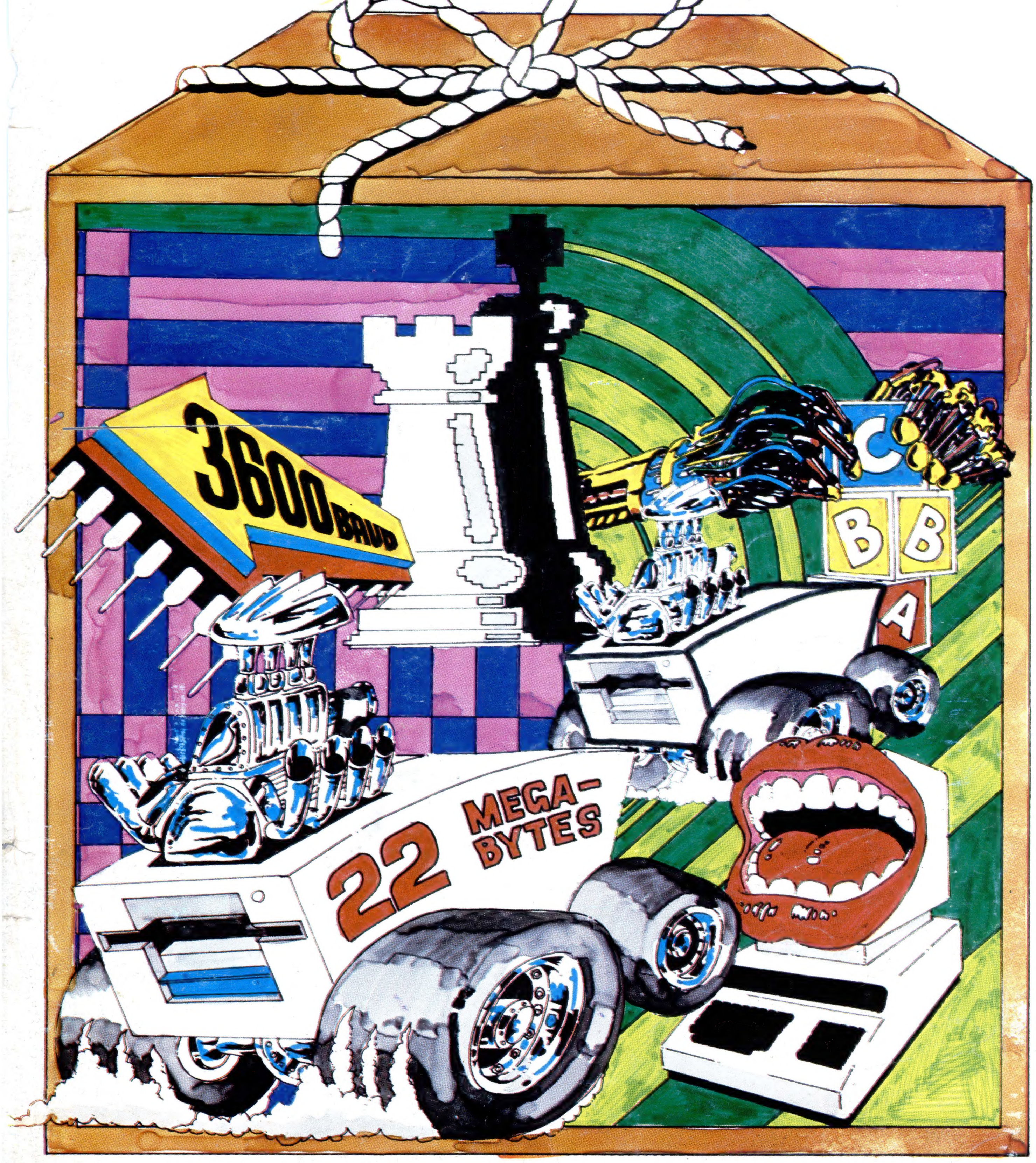
Australian Personal Computer

February 1982
$2.50* N.Z. $3.00*

Australia's leading micro magazine



APPLE III REVIEWED

in APC

Volume 3 No. 2 February 1982

Registered for posting as a Publication Category B ISSN 0725-4415 *Recommended Retail Price only.

ID's WORKSHOP and FIRMWARE CASSETTE UTILITY (by P. Davis) will return next month along with a complete list of winners in our printer competition.

Editors
Sean Howard
Graeme Domm
Technical Editor
Ian Davies
Production Manager
Mike Northcott
Printed by
Lewis Printing,
500 Clayton Road,
Clayton, Vic.

Published by
Howard Productions
P.O. Box 115,
Carlton, Vic 3053
462 Burwood Road
Hawthorn, Vic 3122
Telephone (03) 818 1488
Advertising (03) 818 7508
Telex (AA 30333 AMJ)

Produced under licence from
Felden Productions Ltd.

ADVERTISERS INDEX

$495.00

THAT'S ALL YOU PAY FOR THIS REVOLUTIONARY PRINTER ...

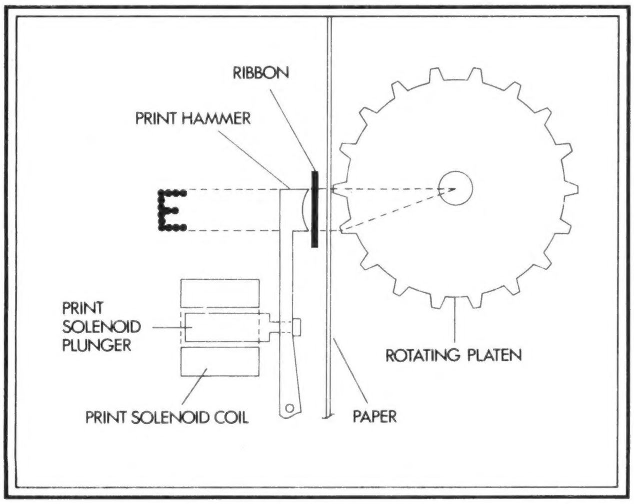
Cat. No. X-3252 P&P $5.50

The Uni-Hammer Replaces Seven . . . or More.

Revolutionary? We don't know what else to call it. An impact printer with a single rugged hammer, rather than the seven or more individual solenoids and print wires found in conventional dot matrix printers.

At an incredible unit price of $495!

Because of the unique Uni-Hammer design, the X-3252 is smaller and simpler than other dot matrix printers yet costs considerably less. Which makes it a natural for the personal or small business user who wants a quality, reliable impact printer at the lowest possible price.



A Product of the Seiko Group

It took a company such as the Seiko group, world's largest watch manufacturer, with vast experience in the design of small, intricate, precision products, to come up with a totally new concept in dot matrix printing.

How the Uni-Hammer Works

The X-3252, which prints both graphics and alphanumerics, uses a rotating platen with protruding splines positioned behind the paper (see diagram). The character or graphics image is created by multiple hammer strikes in rapid succession as the print head advances across the paper. The precision gear train assures exact positioning of the print hammer relative to the splines on the platen, to provide excellent print quality.

A Complete Printer

The X-3252 has features comparable to printers selling for thousands of dollars. These include upper/lower ASCII character sets, ribbon cartridge, 80 columns at 12 characters per inch, adjustable tractor feed, original and 2 copies, 30 characters per second, and full graphics with a resolution of better than 60 dots per inch in both horizontal and vertical axes.

Centronics Interface

The X-3252 DOT MATRIX PRINTER has a Centronics-type parallel data interface and is compatible with System 80, TRS-80, Sorcerer and Apple computers etc.

Fan Form Paper		Replacement Ribbons	
2,000 sheets continuous fan form paper to suit printer			
Cat. X-3254	$27.50	Cat. X-3253	$8.95

EXCLUSIVE TO DICK SMITH STORES

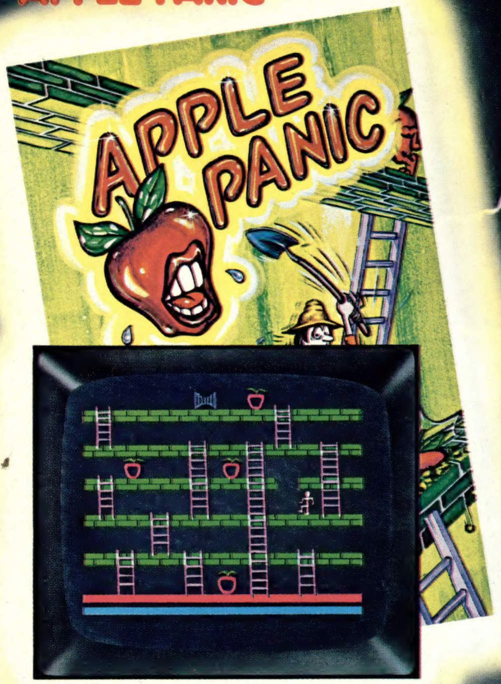
SEE OUR OTHER ADVERTS FOR STORE ADDRESSES.
Available by mail order (P. & P. $5.50 from P.O. Box 321, North Ryde)

bankcard
welcome here

®

DSE/982M/PAI

APPLE PANIC



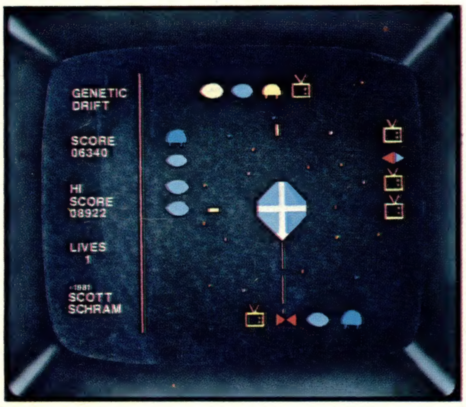
Ladders take you from level to level in this hi-res, fast action Japanese import. Dig holes in the bricks to catch a medley of monsters chasing you about. For all ages. [redacted]

SPACE★QUARKS

The deadly, intricate dances of the space quarks will fascinate you in this hi-res arcade thriller. Careful study is necessary if you are to survive to upper levels. [redacted]

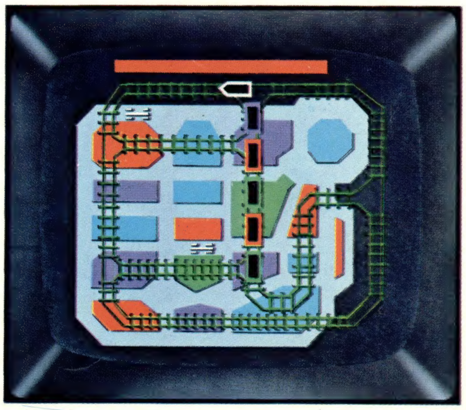
BRØDERBUND SOFTWARE

GENETIC DRIFT



This finger-pounding arcade game requires you to mutate hostile aliens into a benign life form even as they attack from all sides. But watch it! One slip and the friendly ones become deadly once again.

TRACK ATTACK



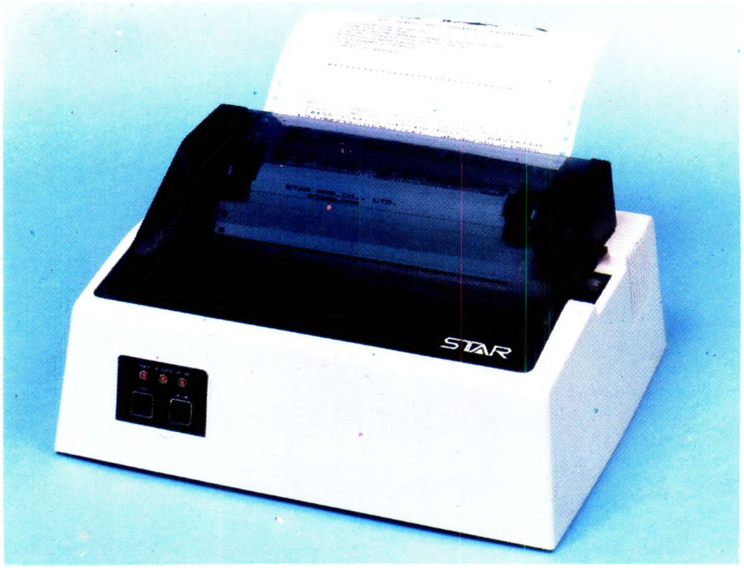
Fast cars and fast trains make this gold heist arcade game hard to beat. To steal the gold on the train while avoiding other cars you need great timing and good peripheral vision. [redacted]

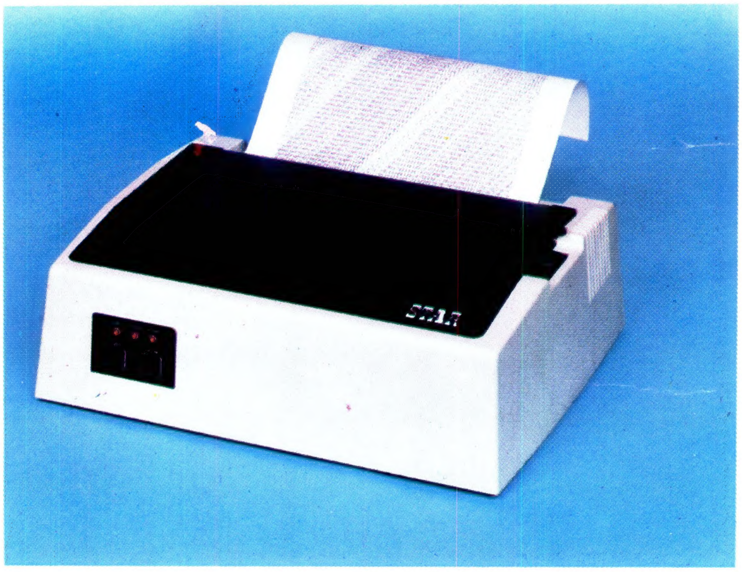
All products require 48K Apple II or Apple II Plus

Please contact Imagineering or your local Apple dealer. P.O. Box 4601, Sydney, NSW. 2001. Telephone (02) 358 3364, 358 3011.

All products require 48k Apple II or Apple II plus and one disk drive, 13 or 16 sector controller.
Apple & Apple II are trademarks of Apple Computer Co

The Greatest Printer Value in Australia





80 CHARACTERS-PER SECOND, BIDRECTIONAL, UPPER/LOWER CASE,

HEADINGS, GRAPHICS, COMPRESSED PRINT, AND IT'S ONLY

$499. * FRICTION/ $565.* FRICTION AND TRACTOR.

SUITABLE FOR USE WITH HITACHI TRS/80, SYSTEM 80 APPLE,

ANY OTHER COMPUTER WITH "CENTRONICS" CONNECTION.

AVAILABLE FROM DEALERS ALL ACROSS AUSTRALIA

OR CALL DELTA, THE DISTRIBUTOR AT: (03) 622 008/9

***Plus Tax if applicable**

TYPE-'N-TALK™ IS T.N.T.

The exciting text-to-speech synthesizer that has every computer talking.

- **Unlimited vocabulary**
- **Built-in text-to-speech algorithm**
- **70 to 100 bits-per-second speech synthesizer**

Type-'N-Talk™, an important technological advance from Votrax, enables your computer to talk to you simply and clearly — with an unlimited vocabulary. You can enjoy the many features of Type-'N-Talk™, the new text-to-speech synthesizer, for just $525.00.
You operate Type-'N-Talk™ by simply typing English text and a talk command. Your typewritten words are automatically translated into electronic speech by the system's microprocessor-based text-to-speech algorithm.

The endless uses of speech synthesis.

Type-'N-Talk™ adds a whole new world of speaking roles to your computer. You can program verbal reminders to prompt you through a complex routine and make your computer announce events. In teaching, the computer with Type-'N-Talk™, can actually tell students when they're right or wrong — even praise a correct answer. And of course, Type-'N-Talk™ is great fun for computer games. Your games come to life with spoken threats of danger, reminders, and praise. Now all computers can speak. Make yours one of the first.

Text-to-speech is easy.

English text is automatically translated into electronically synthesized speech with Type-'N-Talk™. ASCII code from your computer's keyboard is fed to Type-'N-Talk™ through an RS-232C interface to generate synthesized speech. Just enter English text and hear the verbal response (electronic speech) through your audio loud speaker. For example: simply type the ASCII characters representing "h-e-l-l-o" to generate the spoken word "hello".

TYPE-'N-TALK™ has its own memory.

Type-'N-Talk™ has its own built-in microprocessor and a 750 character buffer to hold the words you've typed. Even the smallest computer can execute programs and speak simultaneously. Type-'N-Talk™ doesn't have to use your host computer's memory, or tie it up with time-consuming text translation.

Data switching capability allows for ONLINE usage.

Place Type-'N-Talk™ between a computer or modem and a terminal. Type-'N-Talk™ can speak all data sent to the terminal while online with a computer. Information randomly accessed from a data base cn be verbalized. Using the Type-'N-Talk™ data switching capability, the unit can be "de-selected" while data is sent to the terminal and vice-versa —permitting speech and visual data to be independently sent on a single data channel.

Selectable features make interfacing versatile.

Type-'N-Talk™ can be intefaced in several ways using special control characters. Connect it directly to a computer's serial interface. Then a terminal, line printer, or additional Type-'N-Talk™ units can be connected to the first Type-'N-Talk™, eliminating the need for additional RS-232C ports on your computer. Using unit assignment codes, multiple Type-'N-Talk™ units can be daisy-chained. Unit addressing codes allow independent control of Type-'N-Talk™ units and your printer.

Look what you get for $525.00. TYPE-'N-TALK™ comes with:

- Text-to-speech algorithm
- A one-watt audio amplifier
- SC-01 speech synthesizer chip (data rate: 70 to 100 bits per second)
- 750 character buffer
- Data switching capability
- Selectable data modes for versatile interfacing
- Baud rate (75-9600)
- Data echo of ASCII characters
- Phoneme access modes
- RS-232C interface
- Complete programming and installation instructions

The Votrax Type-'N-Talk™ is one of the easiest-to-program speech synthesizers on the market. It uses the least amount of memory and it gives you the most flexible vocabulary available anywhere.

DICK SMITH Electronics

SYDNEY 290 3377
NEWCASTLE 61 1896
WOLLONGONG 28 3800
CANBERRA 80 4944
MELBOURNE 67 9834
ADELAIDE 212 1962
PERTH 328 6944
BRISBANE 391 6233

Mail Order Centre: PO Box 321, North Ryde 2113. Phone: (02) 888 3200

ANY TERMS OFFERED ARE TO APPROVED APPLICANTS ONLY.

DSE/A122/LM

After the early pioneering days when they were the catalysts for change, data processing departments, like their mainframe suppliers, have been resistant to new technology. In the late 60s, the emerging timesharing services were first taken up by users, with DP dragging along some years later. In the early 70s, the suitability of minis to interactive applications attracted users frustrated with mainframe inadequacies. Many bitter battles with DP departments ensued, although those same departments are now bringing in the super-mini to replace obsolescent third-generation mainframes. When word processing arrived in the late 70s, DP managers again turned their backs and referred the problem to O & M, or those responsible for office equipment. After all, word processors were only replacements for electronic typewriters, weren't they? Many companies now looking at integrated office and data processing systems are regretting DPs' early *laissez faire* attitude towards word processing.

DP departments have fallen behind again with the introduction of micros. Computer managers are finding to their embarrassment that they know less about the subject than users in their companies. The number of DP departments with their own micros is still in the minority. When micros first began to see the light of day, few DPMs noticed them. As the publicity bandwagon for PETs and Apples got rolling, they had little impact on DP. Today, computer people still often associate the word 'micro' at worst with a trivial games playing device and, at best, with a super-calculator. Too few DPMs have reconsidered or altered their computing strategy, set up micro groups or got seriously to grips with the technology that has already turned the computer industry on its head and will progressively do the same to those making their careers in it. It is users who again have pioneered the application of microcomputers and, just in case there are residual feelings that what they are doing is trivial, it is worth pausing to quote some facts.

– *Fact 1:* You can save thousands of dollars on timesharing bills and increase productivity by using micros *now* for budgetting, forecasting, linear programming, PERT, modelling and other management tasks.

– *Fact 2:* For under $17,000, you can purchase *today* a 16-bit desktop computer with 256k RAM (upgradeable to 1 megabyte) plus 10 megabytes of store

PRINTOUT

currently available equipment for anyone considering buying a system.

Write to Jane Yeung, Sybex Inc., 2344 Sixth Street, Berkeley, CA 94710 in the US, or put pressure on ANZ Books in Sydney.

FULL FRONTALS

Gower Smith, who previously distributed Archives Computers through CGF Electronics, among other things, has started up a new company called Archives Computers (Aust). The company will handle both wholesale and retail sales of the computers and ancilliary products in Victoria, and sell nationally through a chain of 22 retailers and OEM'ers. The company has 10 support staff.

Gower said, "Although the Archives III, with its 5.75 megabytes of storage has brought strong interest into the industrial and scientific fields, we intend to make a full frontal attack on commercial fields, with Archives hardware and our locally configured complete office management and accounting software."

Heavy stuff. More than 20 machines have been installed in the past six weeks, and the company has budgeted to support more than 300 systems in Australia by the end of the year. Apparently, the winning features of the Archives are its ability to build from 0.75 Mb to 75 Mb without changing the basic hardware architecture, its 23 programmable function keys, and its locally enhanced version of Wordstar.

The company also distributes and supports Qume daisywheel printers and a range of dot matrix printers.

The address is 162 Clarendon Street, South Melbourne, Vic 3204; Tel: (03) 699 8377.

INDUSTRY SURVEY

The ACS's Software Industry Committee is repeating its software industry survey. It is particularly concerned with getting detailed information about the activity of micro software providers, since this is a growing section of the general software industry.

According to Karl Reed, the chairman of the committee, the nationwide survey is funded by the National Council of the ACS and will be the only authoritative source of data. The results of the survey will be used in submissions to state and federal governments on the industry and its needs. The results should also play an important part in bringing about an awareness of the impact of technology on the Australian economy.

"The computer-related electronics industry is booming in this country, and may be as large as the minerals boom. Surveys such as this are the only means by which the size of this industry can be reliably estimated. Politicians are usually amazed to find that the survey will go to more than 800 companies," Mr Reed said. "Microprocessors will become an increasing part of the computer scene, and supplying appropriate software will become an important industry. As a result the ACS survey must catch as many suppliers of microprocessor software as possible."

He said that both the Department of Overseas Trade and of Industry and Commerce are already taking interest.

Companies marketing micro software, services and packages should write to the ACS-SIC team, c/o Louise Cheung, Dept of Computing, RMIT, PO Box 2476V, Melbourne 3001; or call Ms Cheung on (03) 341 2348.

The Personal Computer Your information window

An extension of your brain
An Apple Personal Computer puts you at the forefront of the technological revolution. Use it well and it will mediate the availabilities of a wealth of conceptual tools, data bases, communication channels and information processes. An Apple will help you simplify a myriad of tasks from ordinary data processing through information retrieval to economic forecasting.

Apple is versatile
Your Apple is an extremely powerful and robust problem solver backed by a comprehensive set of software programmes. Already, over 300,000 Apples are used for business, domestic, educational research and recreational purposes.

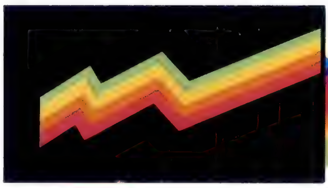
Apple as information processor
An Apple personal computer performs a full range of standard functions like statistics, word-processing, graphics, number crunching, filing, storage/retrieval and cross-referencing.

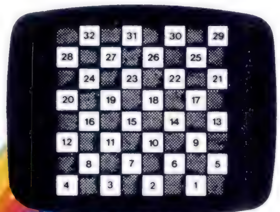
Apple as super-brain
Aside from the more pedestrian functions of retrieving and processing information, Apple performs other sophisticated functions such as econometric modelling, financial forecasting, graphics, topological modelling and trend analysis.

Apple is convenient
Apple is portable. It can be used at home or in the office. You can address it in ordinary English or gain full computer literacy in BASIC PASCAL, COBOL, FORTRAN or PILOT. Full step-by-step training comes with your Apple. If you can use a calculator and speak English, Apple shows you the rest.

Apple as communicator
Apple combines many of the functions of telephones, telexes, citizens' band radios and library access terminals.
It allows you the very latest access to data bases and information services and mail electronically locally or world wide.

Apple as challenger
Apple is fun. From a perfect game of Chess to Star Wars—not only for the children—preparing them for the future in this high-technology age.





Apple is cost-efficient
For roughly the price of a plain paper copier, you get a sophisticated, powerful machine, which not only processes numbers and words, but gives you direct access to a wealth of information through a rich network of data bases and direct communications with other computers.

Mail this coupon for an "Apple Pack" brochure or talk to your local Apple authorised dealer about the specific applications for you.
If the coupon has been removed, send your name and address to:
ELECTRONIC CONCEPTS PTY LTD,
55-57 WENTWORTH AVENUE,
SYDNEY, NSW 2000.

apple computer®

Mail to: PPCB374B
ELECTRONIC CONCEPTS PTY LTD,
55-57 WENTWORTH AVENUE, SYDNEY, NSW 2000.
Please send me a free "Apple Pack" brochure.

NAME ____________________
ADDRESS ____________________
____________________ POSTCODE ________
PROFESSION ____________________

APPLE APPLE = PERSONAL COMPUTER PERSONAL COMPUTER = APPLE APPLE

PRINTOUT

CP/M LIBRARY

The Automated Office, a Sydney software company, has over 50 volumes of software from the New York CP/M Users Group. The software is in the public domain and many programs contain the original source code which the user is free to modify or distribute but not sell. There is a charge for floppy disks and postage.

The library contains an international collection of software, including application programs, system utilities, communication programs, compilers and languages. Each volume contains the comments of reviewers and notes on installation and operation.

The catalogue and further information is available for $4 from The Automated Office, PO Box 490, Chatswood, NSW 2067; Tel: (02) 411 1892.

COMPUTERS GALORE

Apart from some metaphysical confusion about tomorrow being today, Steve Colman seems to be on the right track with his new retail shop, Computers Galore.

His prognosis: "With the extremely low costs of complete systems at $30 to $40 per week if leased, every business can afford a computer and word processor. The combination has a capability which is hard to believe. In addition to a superior system for writing letters, direct mail and even typesetting and the complete accounting function, there are now programs available for superceding filing systems, card systems, and ready made programs for budgeting, invoicing, reporting, cash book and banking. Also the same computer can now be used to connect with other computers here and overseas."

I believe it.

The new shop, at 99 Military Road, Neutral Bay, NSW, will cater for small business and tradesmen. In addition to a range of microcomputers, it also stocks calculators, plain paper copiers and computerphones, and has a photocopying, instant printing and word processing service. The telephone number is (02) 908 2355.

WESTERN MEDIA

People who live in the capital of Court's (almost ex-) Kingdom now have a handy Magmedia office. It will stock the full range of products including Verbatim flexible diskettes, Wabash computer tape, Magmedia ribbons, disk packs, cartridges, furniture and other accessories. The address is 252 Sterling Street, Perth, WA 6000; Tel: (09) 328 3311.

AND AGAIN

Australian Computer and Telecommunication, a division of API, has begun manufacturing 5¼ inch Winchester controllers for the Australian and UK markets.

They can be used with any of the popular Winchester drives, with capacities of up to 18Mb. ACT is making CP/M software available for the Tandy, North Star, Superbrain, Heathkit, Industrial Micro Systems S100, with Apple and Dick Smith systems coming soon.

The cards are available in two different configurations: the single controller card with separate host interface, or the dual S-100 card set. Each controller set uses the HOP digital data separator, removing all adjustments from the controller while decreasing the separator error resolution.

ACT says its new complete system, including 5Mb hard disk controller and power supply, will be available by mid-March in production quantities at a retail cost of $3,400. ACT lives at 75 Willoughby Street, Crows Nest, NSW. Tel: (02) 439 6300.

EL ZX81

According to the press release, Sydney siders have had difficulty obtaining supply and support for the el cheapo but el interesting ZX81 computer. Computer Galerie has been appointed the North Shore distributor.

What's more, the boss, David Diprose, has put together a Teach Yourself Programming package to go with it.

PRINTOUT

Computer and course cost $295 including freight.

Computer Galerie is at 66 Walker Street, North Sydney, NSW 2060; Tel: (02) 929 5497.

MASTER PLANNER

Informative Systems has a new Cromemco financial planning and general purpose numeric analysis package called Planmaster. It provides automatic spreadsheet analysis for financial planning cost accounting, sales forecasting, production planning, cash flow analysis, scientific data evaluation, and hundreds of other applications. Data entry is via direct cursor positioning to the desired location, with decimal positioning done automatically.

A plan can include up to 1010 lines or up to 130 columns, limited by a maximum of 12,000 entries. Each line or column can be given a user-defined label, while mathematical relationships between the entries may be defined in terms of these labels.

Available on 5 inch or 8 inch diskette for $395, from Informative Systems, 337 Moray Street, South Melbourne, Vic 3205; Tel: (03) 690 2899.

WINCHESTER

More from Warburton Franki for the Zenith micros. An 8 inch Winchester disk drive with floppy disk back up has been introduced by Zenith Data Systems for its business machines.

The non-removable Winchester in the new Z-67 increases the storage capacity to almost 10 million bytes, with the 8 inch floppy diskette back-up providing an additional 1 million.

The floppy is provided for back-up, data interchange and portability of programs and data. It is compatible with the industry standard IBM 3740 format and will record in single or double density; either single sided or double sided. The diskettes are also compatible with the Z-47 dual 8 inch system, which has been around for a year.

The floppy disk drive sits alongside the Winchester in a cabinet. A switching power supply is built in. The new drive connects directly to the back of the Z-89 or Z-90 micros using a flat cable.

LEARN WITH ZEST

Zenith Education Systems will be running two courses beginning in early March. "Computer and Computing – Getting Started" it is for complete novices; and "Basic Programming" will include graphics and sound generation.

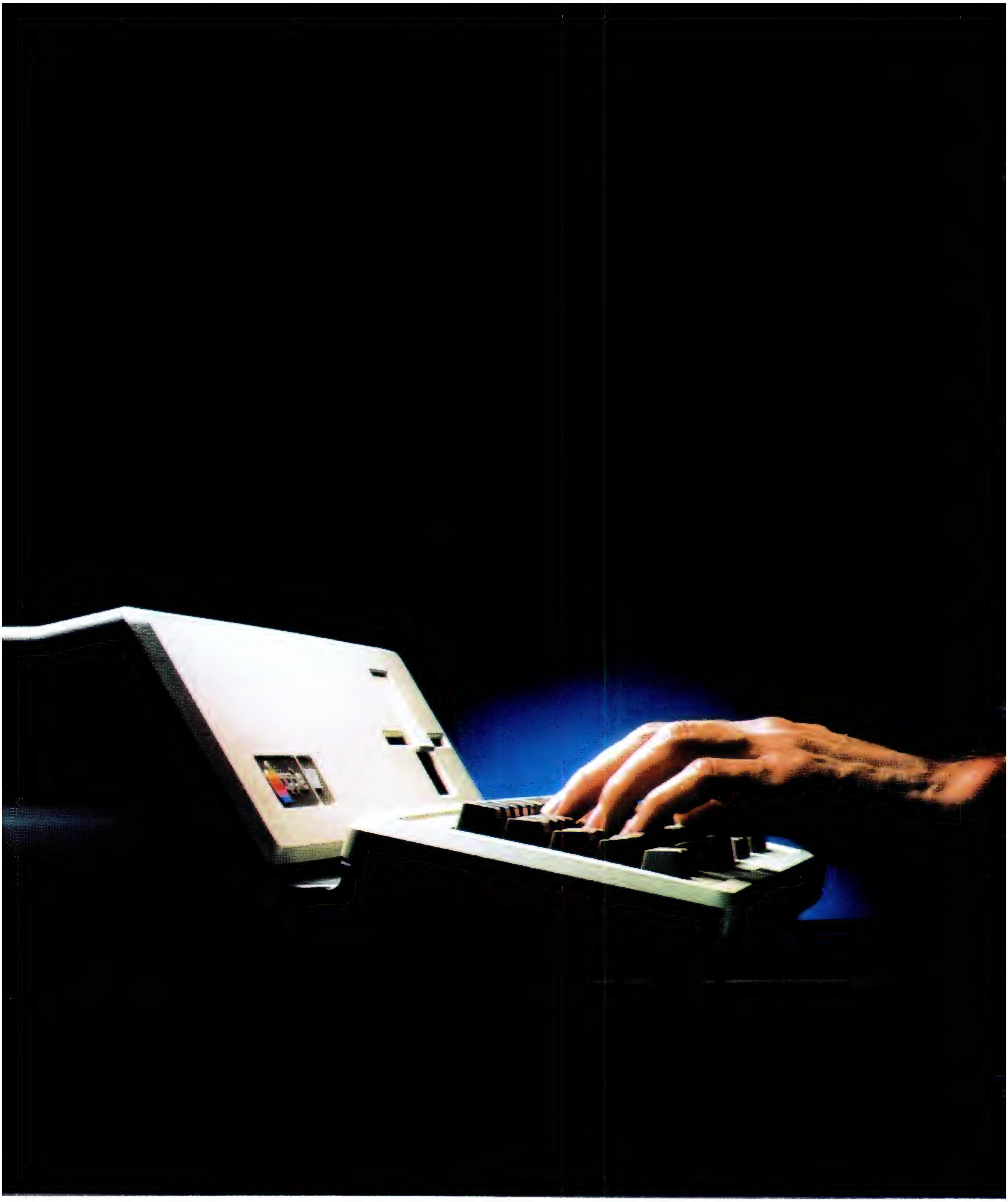
There will be practical work and class sizes will be restricted to one instructor for five students.

For further details, write to Zenith Education Systems, P.O. Box 505, Bankstown, NSW 2200; or ring (02) 708 3140.

P.S.

The latest rumours about Hitachi are incorrect. Contrary to the opinions in the APC art room, no left hand versions of the Peach are being manufactured. For once my source is both authoritative and emphatic (!)

The Zenith Data Systems Winchester disk drive.



APPLE III

REVIEWED

by Ian Webster.

The Apple III first appeared as an ever changing set of rumours in 1979, when people realised that Apple's considerable development effort was about to produce a new machine. The machine was released at NCC 81 and quickly gained a reputation for being bug-ridden. Consequently, the machine was put on hold while the problems were fixed. Most of them were associated with chips coming loose from the sockets in the motherboard and clock chip stability.

The release of Pascal III and the Profile hard disk system provided Apple with an opportunity to relaunch the III and to express their satisfaction with the product.

Steve Jobs has said that Apple intends to build a range of machines exploring the dimensions of the personal computer, and this has been the basis of Apple's design philosophy. These machines are based on large RAM based computers supported by sophisticated operating environments. This evaluation was written during a Festival of Sydney workshop. People with experience in video, audio and photography were being introduced to computers as a creative medium. The experience of the workshop really confirmed the importance of Steve Jobs observations.

THE STANDARD MACHINE

The Apple III is a 6502B based machine with 128k of RAM memory. There is a built-in 5.25 inch floppy disk drive. The machine has an RS232B port, two joystick interfaces, video and audio ports, a Silenttype printer interface and four expansion slots. The system is packaged with a separate monitor: either colour or green screen. The SOS operating system, Business Basic and Visicalc are supplied with the machine. The expansion options are to daisy chain up to three floppy disk drives from the built in drive, the Profile hard disk and Apple Pascal III.

HARDWARE

The Apple III has a cast aluminium case with an extended keyboard. The motherboard is under the case. There is a moulded plastic cover over the built-in disk drive, the power supply and a space for four expansion boards. The computer is fairly heavy and sits firmly on the table.

The keyboard extends from the front of the computer, and could easily have been detachable. The keyboard has 61 keys and a 13 key keypad. All 128 ASCII characters can be generated and all keys have auto-repeat. (The four cursor keys have two speed auto-repeat.) The keyboard is an improvement on the Apple II keyboard, but the keys still feel a little insecure.

A 5¼ inch (140k) floppy disk drive is built into the lefthand side of the case. The disk controller is on the motherboard and can accommodate three daisy chained drives.

A 2 inch speaker driven by a six bit DAC is mounted in the center of the case.

The motherboard is a 15 inch x 10 inch board that is only accessible from the bottom of the case. The board contains all chips except for the RAM which is contained on a piggyback card that has 32 16k bit RAMs and 16 32k bit RAMs. The board can take 256k RAM.

There is only 4k ROM on board and a number of unidentified proprietory chips. An empty socket is intended for the clock chip, and an empty space for the clock battery. The board is well designed with only one wire jump and has a number of undocumented plugs that may provide access to features of the system.

All interface connectors are on the back end of the motherboard. The expansion sockets are accessed through the top of the computer. Video signals available on the board are PAL B&W, PAL composite video and true RGB video. A variety of text and graphic display modes are available. The standard text display is 80x24 characters, with options for 40x24 black and white and 40x24 with 16 foreground and background colours.

Graphics modes are 280x192 and 560x192 in B & W. Color modes are 280x192 with 16 colours and a composite 140x192. Colour is displayed as a grey scale on a B & W monitor.

The board has several memory management switches to support a 64k RAM space with the 6502 processor. The boot ROM is mapped into $F000-$FFFF and is switched out of memory space after system boot. The memory mapped I/O space at $C000-$CFFF can also be switched out of address space and replaced with RAM and system supports multiple stacks. The Environment register at $FDFF, used to reflect the state of the system, is a good example of the type of hardware environments that can be established.

ENVIRONMENT REGISTER

BIT	USE	CLEAR	SET
7	Clock Speed	2 Mhz	1 Mhz
6	$C000 switch	RAM	ROM
5	Video output	Off	On
4	RESET key	disable	enable
3	Protect $C000	On	Off
2	Stack	reloc	$100
1	$F000 switch	ROM A	ROM B
0	$F000	RAM	ROM

The hardware is an innovative design to provide an environment that will support more than 64k of memory and integrate standard peripheral devices into the motherboard.

The Apple III was designed to provide a standard set of peripheral devices on the motherboard and to support a hard disk. Apple began to develop a hard disk system for the Apple III in 1980, based on a Seagate 5Mb drive, and the PROFILE hard disk system has just been released.

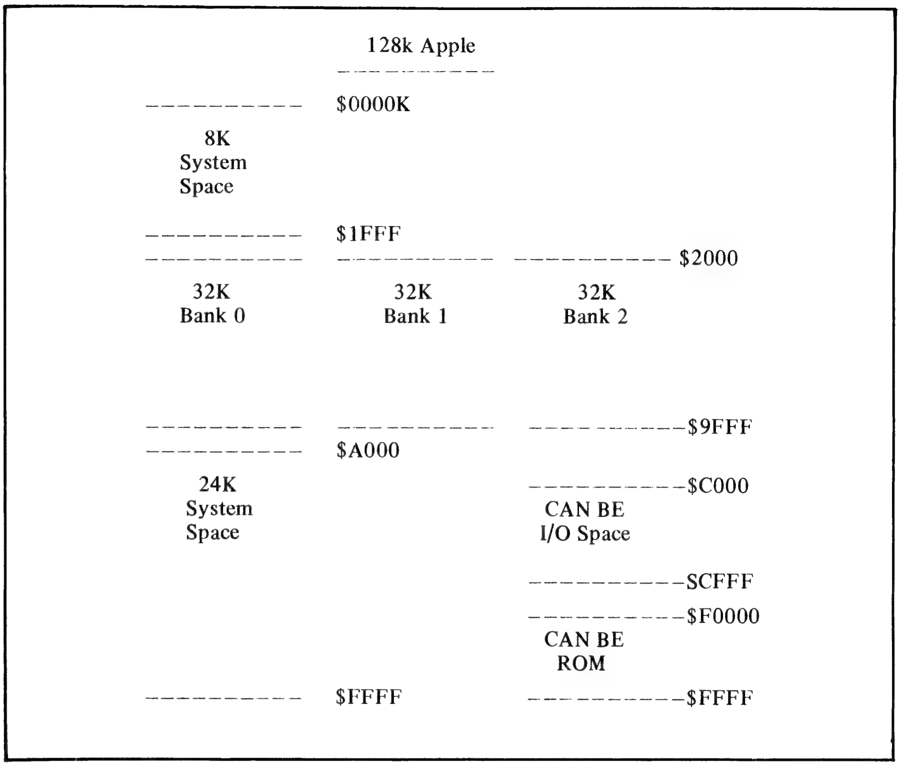
The disk is enclosed in a small moulded plastic case (15" x 8" x 4") in regulation Apple colours. Installation of the disk requires that a driver PROFILE be configured into the system and the interface card plugged into the Apple III expansion slot so that the disk is then fully supported by the SOS. A good feature is the quietness of the drive, even during extensive disk searching.

There is a parallel card for the Apple III and other cards are being developed. Overall, the expansion bus is very similar to the Apple II bus: small interface cards that do not use a lot of the bus will work in the Apple III system. Microsoft is developing a Z-80 Softcard. The Keyboard Company has developed a joystick and M&R Enterprises have released a RGB video interface box.

SOFTWARE

The system software for the Apple III consists of 4k boot ROM and the SOS operating system.

The motherboard ROM contains the system boot code and a series of diagnostic tests that are performed after every boot. The diagnostics test RAM, ROM, I/O ports and the memory management hardware. The ROM also contains an undocumented monitor with similar functions to the Apple II



TECHNICAL SPECIFICATIONS

Size	17.5" x 18.2" x 4.8" Cast aluminium base with plastic moulded cover
Weight	26lbs
Processor	6502B with extended addressing hardware
Clock Speed	2Mhz peak, 1.4Mhz average
RAM memory	128 k
ROM memory	4 k
Disk Storage	One 5.25 inch floppy disk (140k)
Keyboard	74 Keys with auto-repeat, generating the full ASCII set
Screen	Text
	40 x 24 Black & White
	80 x 24 Black & White
	40 x 24 16 foreground and background colours
	All characters are software definable
	Graphic
	280 x 192 Black & White
	280 x 192 16 foreground and background colours
	140 x 192 16 colours
	560 x 192 Black & White
Video Output	Black & White
	Composite colour
	RGB pure video
	Signals Pal standard
Audio output	2 inch speaker with a 6 bit DAC
Serial I/O	RS232B port
Joystick	2 joystick interfaces
Printer	Apple silenttype interface
Expansion	4 50 pin expansion slots

PROFILE HARD DISK

Disk	Seagate drive – 5M formatted data
Surfaces	4
Heads	4
Tracks/Surf	153
Sectors/Trk	16
Bytes/Sector	532
Rotation	3600 RPM
Seek time	95 msec
Data trans	5 Megabits/sec

monitor. Memory can be moved, verified, displayed and searched. The monitor can also Read and Write disk blocks.

SOS provides a complete environment for all software on the Apple III. The system has combined the Apple DOS user command interface with the structure of the UCSD operating system and UNIX style hierarchical directories. A uniform interface to the system is provided for all language implemented on the Apple III.

SOS provides extensive high and low level calls to the system.

It consists of KERNEL, which has the low level interface to the system hardware, a DRIVER file that contains the interfaces that all peripheral devices configured into the system, and a P-code interpreter.

SOS has five component interfaces. The first is the file management interface. All devices in the system can be treated as clock or character structured files. A hierarchical directory structure is used where each device has a root directory that may contain both files and other directories. The location of a file is specified by the pathname of the file. The root directory must be specified followed by the other directories until the directory that contains the file is reached. A file called DEMO.CODE that is in a directory called MYDISK on the Profile hard disk would be accessed by the pathname:– PROFILE/MYDISK/ DEMO.CODE. This feature is one of the hardest for Apple II users to get used to, but provides a powerful facility for the management of files on large storage devices.

A drawback is that SOS, in its provision of a universal interface for all languages, has a proliferation of filetypes. It supports 13 filetypes, from SYSTEM to UNKNWN.

The second interface is the device management interface that provides access to the drivers. The drivers accept control codes for the operation of the device or to report on status enquiries. The CONSOLE driver controls cursor position, screen window size and text colour. The 128 byte type ahead buffer and the software definable character sets are also controlled by the CONSOLE driver. The keyboard can also be remapped through the device control interface. The GRAPHIX driver provides control over the colour options table and the type of point plot on the screen. The RS232B interface can specify baud rate, data format, buffer size and a choice of no, XON/XOFF or ENQ/ACK communication protocols.

The third interface is the memory management interface that manages the allocation of memory in the system. Switches control the bank select, I/O space select and the ROM select. Memory management also controls stack relocation and the management of memory in the system.

Interrupt management is largely handled by SOS, although a user interrupt mechanism has been implemented that preserves the priority of SOS peripheral interrupts.

The utility interface provides access to the software clock and joystick ports in the system.

The SOS software includes a utilities disk that is an enhanced version of the Pascal Filer. The system support software is written in Pascal. This utility allows manipulation of disks and files, format disks and generation of a new system.

Apple III Pascal is an enhanced version of Apple UCSD Pascal 2.1. SOS has lessened the severity of the UCSD operating environment. Several additions have been made to the system including an OTHERWISE clause for the CASE statement, Bytestream and Wordstream datatypes, conditional compilation and the implementation of the IEEE single precision floating point standard. The general parameters of the system have been enlarged to utilise the available memory space. A compiler option ($SETC APPLE:=2) will generate code that is compatible with Apple II Pascal.

Essentially, the Basic has been written to the Microsoft V5.0 standard with several extensions. The language has adopted the Microsoft format for Disk instruction (OPEN #1 AS OUTOUT). The most interesting feature of the language is the implementation of LONG INTEGERS with 19 digits for financial programming, a powerful PRINT USING statement, indented program listings to highlight structures, and ON KBD and ON EOF# statements.

There are no PEEK, POKE and CALL commands. The interface between Basic and machine code is contained in the INVOKE and PERFORM commands. INVOKE will load a machine code file into memory. SOS handles all memory management details and decides where to put the file. PERFORM will call the routine passing a parameter list via the stack. PEEK and POKE have no place in an environment where the system is capable of managing the environment and where there is enough control with system calls to interrogate the system. If PEEK and POKE functions are required then a machine code routine can be written to provide the information.

A separate graphics module provides commands to Draw images, points and lines and to Fill screen area and control colour selection.

Business Basic should have been available for the Apple II years ago, and will make Apple II programmers realise the inadequacy of the Applesoft Basic implementation for serious programming.

Apple has released an enhanced version of VISICALC for the III and has a word processor called Word Painter almost ready for release. Several Special Delivery Software packages have been released, including Apple Writer. There are at least six other word processing programs available for the Apple III.

Enhanced versions of Desktop Plan and PFS are also available. The PFS dealer disk includes an impressive demo for the Profile that contains all of the Skarbeks current software directory.

Paul Lutus has also produced versions of his assembler, ALD III and Forth, TransForth III for the machine.

The manuals are a delight to use, explaining the operation of the system in clear and patient style. The documentation assumes that the user is reasonably computer literate and there is little tutorial material. The specification of Business Basic and Pascal is excellent, but I get the impression that Apple has omitted a lot of technical information in an attempt to dissuade people from tampering with SOS.

The manuals have been written from the programmers point of view and provide all the information necessary to use the system.

APPLE II EMULATION

The Apple II emulation mode converts the Apple III into an Apple II. The clock speed reverts to 1 Mhz and memory is configured to a 48k Apple. The emulation disk can provide Integer or Applesoft in ROM and the Apple III serial port can be configured to emulate an Apple high speed serial card or a communications card. The configuration cannot be changed without a reboot of the emulation disk.

A wide variety of Apple II programs were tested under emulation mode including several games which are known to exploit features of the Apple II. All programs executed without any problems. Problems could occur with the game socket because the Apple III emulates the paddles using joystick ports. If a program does not use a standard paddle driver then the Apple III joystick may not respond correctly.

Not all of the features of the Apple III keyboard can be used in emulation mode as the keyboard is remapped and some keys do not generate ASCII codes that correspond to that key.

CONCLUSION

The power of the system is not in the hardware but in the complete machine environment.

The major achievement of the system is compatability with Apple II while building a software system that provides independence from the hardware design of future Apple computer systems.

The machine will be a delight for programmers, perhaps a little daunting to the home hacker and a powerful work station for users who require a computer system to assist with their work. Anyone with an information management application that requires a small hard disk should consider the Apple III.

CUBIC SPLINE CURVE FITTING

MJ Parrot explains a technique for fitting a smooth curve to complex dataplots

Before your eyelids begin to droop uncontrollably, I must hasten to explain that Cubic Spline Fitting has nothing whatsoever to do with Mr Rubik, less still with the reassembling of his cubes. Cubic spline interpolation is a mathematical technique which does for a curved line what linear regression does for a straight line graph. Even if you are not a mathematician, ponder a while on whether or not you have ever wanted your program to display results as a graph; not a messy scatter of points but a line or smooth curve, which enables you to predict values other than those plotted. These may be sales figures or number of shark attacks per year, laptimes of a racing bike by percentage of a fuel additive, soccer scores by phase of the moon – there are many areas outside of pure mathematics and statistics where the correlation between two related phenomena can best be understood by graphical display.

Anyone who has been involved in a discipline where experimental or sample data have to be presented in graphical form will probably be familiar with linear regression (LR) analysis. The data points are plotted on graph paper; they appear to show a trend, but they by no means fall into a neat straight line. In fact, a multitude of straight lines could be drawn which pass close to most of the points. LR analysis finds the single straight line which is optimally close to the most points; in other words, the line which best represents the trend of the data.

The actual method used is based on minimising the difference between the sum of squares or products of data coordinates and the squares or product of their means; it yields as well as the equation of the best fit line, a coefficient of correlation and variance (a measure of the goodness of fit). The method can be found in any statistics textbook and programs are widely available for scientific calculators as well as for micros. The 'least squares' method can be extended to fit some simple curves to data, namely exponential, logarithmic and power regression. But, by and large, these only work well for a continuously increasing or decreasing function. If the best graph for your data has several turning points, then the far less widely known technique of cubic spline interpolation can fit a series of cubic curves to the points. The following program is for Apple II, and the relevant portions could easily be lifted out for inclusion as a subroutine in a larger program or suite of programs, in order to present results as a smoothly curved graph.

The program allows the plotting of up to 50 points on Apple's high resolution screen and allows the user to remove one or more points. This is useful if one point is so far off the curve as to be suspect.

Essentially a cubic spline consists of cubic equations knotted together at the datum points. These equations are mutually dependent in that on either side of each datum point they have the same x & y values (naturally), the same slope, and the same curvature. Briefly, for each cubic equation, interpolated between points with x-values x_i and x_{i+1} the equation has the form:

$$y = a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)3$$

The program is written as a collection of subroutines which may easily be added to or changed. (It is worth noting that several of the routines could be used as elements in a linear regression program.) The sort routine (line 190) is a fast sorting routine for the number of points likely to be involved and is really only used to find the minimum (left in A(0)) and maximum (left in A(N–1)) values of the points 0 to (N–1).

In the program the linear parameters a_i, b_i, c_i, d_i have been calculated and stored in the arrays A1(I), B(I), C(I), D(I). Note that early on in the calculations the arrays B, C, D are used for other parameters which become redundant. Having calculated the spline parameters the program draws the curve. In both programs the axes are arbitrarily drawn, although it is quite easy to scale them and also to label them using a shape table of characters.

With the cubic spline technique, the order of data input is obviously important as pairs of data are used to calculate the parameters, therefore some care is needed in entering the data. Also, to facilitate curve drawing and to lessen memory overheads, some assumptions have been made. These are that the x-values increase from one point to the next and that the curve generated is not going to go wildly off the screen. If these points are not adhered to, the program will not crash but will draw some 'odd' shapes. Thus the program will not draw a spiral through points which lie on a spiral although the parameters have been correctly calculated. A more wide-ranging plotting technique could easily be implemented if the user desires.

The cubic equations generated may be used to 'read' a value from the graph or may be used to draw the first derivative of the curve since parameter B is the value of the slope at each point and this can be plotted and a second cubic spline can be interpolated.

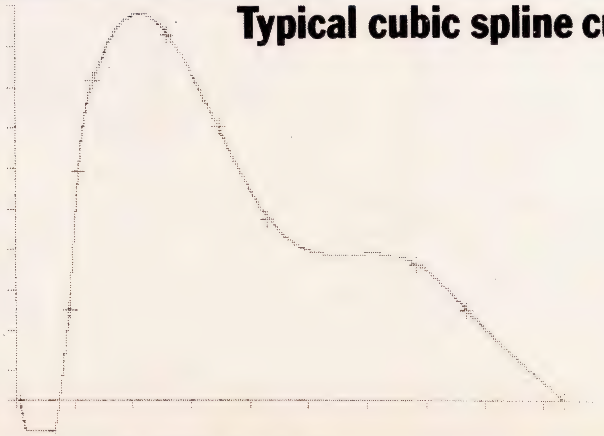
The output formatter of the program merely aligns output within a field and can easily be dispensed with. If anyone wants to use it in other programs, note that the subroutine requires F(1), the number to be output; F(2), the number of decimal places to be printed; F(3), the size of the field in which to print and an array F$(35).

Conversion to other Basics

The major difficulty in converting this program to run on other micros will centre around their plotting capabilities, but I don't see why it should not be possible. Points to note are:

1. HOME. . . clears the screen & returns the cursor to top left-hand corner;
2. CALL-868. . . clears from cursor to end of the line;
3. POKE 32,4. . . moves the left-hand side of the text window to column 5. This is used to indent a table. POKE 32,0 restores the full text window;
4. LOMEM: 16384. . . used to set the start of variable space in order to protect the high resolution page;
5. VTAB & HTAB. . . are respectively vertical & horizontal tabbing commands used outside of print statements;
6. HPLOT x,y. . . plots a point at the coordinates x,y on a grid where x runs from left to right with values 0 to 280 and y runs from top to bottom with values 0 to 160 (on the part of the screen seen in the program);
7. HPLOT TO x',y'. . . plots a straight line from the last point plotted to the point x',y'.

Typical cubic spline curve fitting



MICROCOMP TAKES OFF

MICROCOMP began in October, 1979, with just Nicki and Bill and the Commodore Product. In April, 1980, we moved into the City and the following October, John Phillips joined the team as a Programmer. Errol the Cat moved in during July, 1981. Now into our 3rd year, Andy Johnston has joined as our Systems Engineer and Robert Mitchell as a Trainee Programmer. Because of extra staff and more products to display, MICROCOMP has taken extra space on the 3rd Floor of 561 Bourke Street. This area is for an additional Showroom handling the new VIC 20, the hobbyist, educational and personal Microcomputer market, leaving the 4th Floor Showroom for the Business orientated Microcomputer market.

All this has been due to the loyalty of our existing customers and their continued business. The success of the Commodore Range, their reliability and good value, have made MICROCOMP the leading Commodore Dealer in Melbourne.

Our new Showroom officially opens on Tuesday, 2nd February, and we are having an OPENING SALE to celebrate. So please come and see our Microcomputer range, our vast selection of Microcomputer books and our various Microcomputer related products, such as High-Resolution Graphics, Programmers Toolkits, RS232 Test/Patch Sets, KC NETKITS, Softbox CPM Adaptor, Corvus Hard Disks, Epson MX80 and MX100 Printers, the American Source Database and more.

We pride ourselves on our friendly and efficient service. Our expansion does not mean less individual attention to our customers. We can now offer more services and keep up with our ever increasing workload.

SO COME AND SEE THE MICROCOMPUTER SPECIALISTS AND BE ASSURED OF GOOD, KNOWLEDGEABLE SERVICE.

master charge VISA

MICROCOMP
MICROCOMPUTER SYSTEMS DESIGNERS

B.S. MICROCOMP Pty. Ltd.,
4th Floor,
561 Bourke Street,
MELBOURNE, 3000.
Tel: 614-1433/614-1551.

THE SMALL COMPUTER LARGE ENOUGH FOR 8 FULL-SIZE ADULTS.

Small computers aren't kid's stuff anymore.

Micromation's M/NET™ system delivers big computer power to small computer users. So do a lot of other products in single user systems.

But M/NET has a feature just right for growing companies. And for large companies with special applications. It's called upgradeability.

Because when a company grows, so does its need for computer power. And that's where M/NET moves out ahead. For each user terminal you add, M/NET adds a processor and 64K of memory. Not just out to 3 or 4 users. But all the way up to 8 processors for 8 on-line terminals.

So M/NET users don't suffer from the response-time delays common in other multi user, micro-based systems.

The M/NET system gives you the flexibility of a modular architecture, plus CP/M* software compatibility. And your choice of data storage: floppy disks and Winchester-type drives.

Here at **Microprocessor Applications P/L** we've spent four years building a professional company.

We provide the technical backup, the service, and the support that business users—and business computer dealers and OEM's—demand.

If you're serious about using computers to help your business, call us for the name of your nearest Micromation distributor. And if you're a computer dealer that's serious about expanding your market coverage and customer base, call us for full details on our complete M/NET dealer support program.

We'll show you why M/NET is the small computer large enough for you.

MICROMATION M/NET™

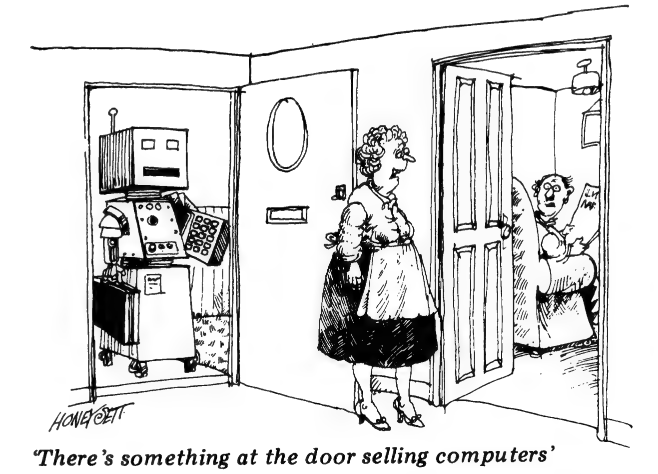
IT'S ONLY SMALL ON THE OUTSIDE.

Microprocessor Applications P/L

Microprocessor Applications P/L Maskells Hill Rd. Selby. 3160. 754 5108, 754 7233.

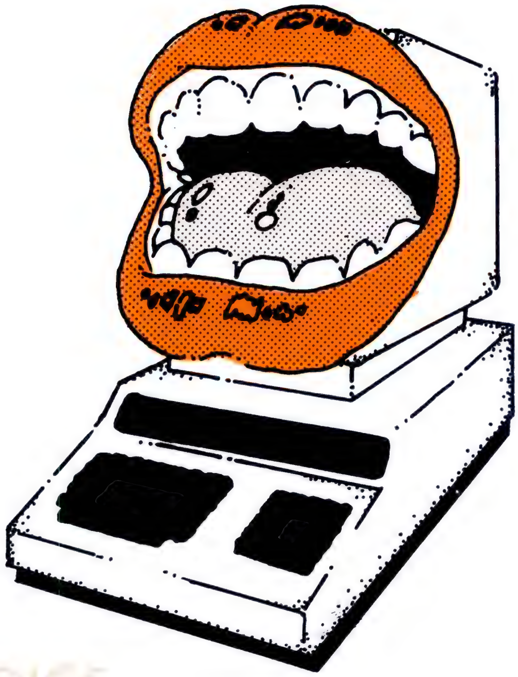
CP/M is a registered trademark of Digital Research

```
1  LOMEM: 16384: POKE 232,009: POKE 233,66
5  DATA   "1","2","3","4","5","6","7","8","9","0"
10  DIM P(1,30),F$(35),F(10),A(30),X(32),Y(32),A1(30),B(30),C(30),D(31),V(30),W(30)
20  GOTO 10000
24  REM

SPACE BAR

25  VTAB 24: HTAB 7: FLASH : PRINT "PRESS SPACE BAR WHEN READY";: NORMAL : GET T$: RETURN
39  REM

OUTPUT FORMATTER

40  FOR J9 = 1 TO F(3):F$(J9) = " ": NEXT J9: IF F(2) > F(3) - 2 THEN  GOTO 120
45 F(2) =  ABS ( INT (F(2) + .5)):F(4) = F(3) - F(2):F(5) =  ABS (F(1)) + .5 * 10 ↑ ( - F(2)):F(6) =
 0:F(8) = 0: IF  INT (F(5)) = 0 THEN  GOTO 85
50  IF F(6) > 0 THEN F(5) = F(5) / 10
55 F(6) = F(6) + 1: IF F(6) > F(3) - F(2) - 2 THEN  GOTO 120
60  IF  INT (F(5)) > 9 THEN  GOTO 50
65  FOR J9 = F(6) TO 1 STEP  - 1: IF  INT (F(5)) = 0 THEN F(5) = F(5) + 10
70 F(8) = F(8) + 1:F$(F(4) - J9) = F$( INT (F(5)) + 25): IF F(8) > 6 THEN F(5) = 0
75 F(5) = 10 * (F(5) -  INT (F(5))): NEXT J9: IF F(8) > 6 THEN  GOTO 105
80 F(5) =  ABS (F(1)) + .5 * 10 ↑ ( - F(2))
85  FOR J9 = 1 TO F(2):F(7) = F(4) + J9:F(5) = 10 * (F(5) -  INT (F(5))): IF  INT (F(5)) = 0 THEN F(
5) = F(5) + 10
90 F(8) = F(8) + 1:F$(F(7)) = F$( INT (F(5)) + 25): IF F(8) > 6 THEN  GOTO 100
95  NEXT J9
100  IF F(2) > 0 THEN F$(F(4)) = "."
105  IF F(1) < 0 THEN F$(F(4) - F(6) - 1) = "-"
110 F$(F(3) + 1) = "":F$ = "": FOR J9 = 1 TO F(3):F$ = F$ + F$(J9): NEXT : IF  ABS ( VAL (F$)) >  =
1 THEN F$ = " " + F$: RETURN
115 T$ =  LEFT$ (F$,F(4) - 1):Z$ =  RIGHT$ (F$, LEN (F$) - F(4) + 1):F$ = T$ + "0" + Z$: RETURN
120 F$ = "": FOR J9 = 1 TO F(3):F$ = F$ + "$": NEXT : RETURN
180  REM

SORTING

190  FOR J = 1 TO N
200 I = J - 1
210 A = A(J)
220  IF A >  = A(I) THEN 260
230 A(I + 1) = A(I)
240 I = I - 1
250  IF I >  = 0 THEN 220
260 A(I + 1) = A
270  NEXT J
280  RETURN
350  REM

AXIS PLOTTING

360  HPLOT 3,0 TO 3,159: HPLOT 0,156 TO 279,156
370  FOR I = 0 TO 159 STEP 16
380  HPLOT 0,I TO 3,I
390  NEXT
400  FOR I = 3 TO 279 STEP 28
410  HPLOT I,156 TO I,159
420  NEXT
430  RETURN
630  REM

POINT REMOVAL

640  VTAB 23: PRINT "DO YOU WANT TO REMOVE A POINT? Y OR N?"
650  GET T$: IF T$ = "N" THEN  RETURN
660  HOME : VTAB 21: PRINT "THE BOX WILL MOVE TO EACH POINT IN TURN"
670  PRINT "IF REMOVAL WANTED TYPE R, ELSE K"
690  FOR I = 0 TO N: GOSUB 870
720  GET T$: IF T$ = "R" THEN  GOSUB 780
730  IF T$ = "K" THEN  HCOLOR= 0: GOSUB 870
740  HCOLOR= 3
750  NEXT
760  RETURN
770  REM

ARRAY SHUFFLE AFTER POINT REMOVAL

780  FOR K = I + 1 TO N
790 X(K - 1) = X(K):P(0,K - 1) = P(0,K):Y(K - 1) = Y(K):P(1,K - 1) = P(1,K): NEXT : FOR K = N TO N +
 1:X(K) = X(K + 1):Y(K) = Y(K + 1): NEXT :N = N - 1: POP : POP : POP : GOTO 10010
860  REM

BOX PLOTTING

870  HPLOT X(I) - 5,Y(I) - 3 TO X(I) + 1,Y(I) - 3 TO X(I) + 1,Y(I) + 3 TO X(I) - 5,Y(I) + 3 TO X(I)
- 5,Y(I) - 3: RETURN
5005  REM

PLOT POINTS

5010  FOR I = 0 TO N:X(I) =  INT (5.5 + SX * (P(0,I) - X0)): NEXT
5020  FOR I = 0 TO N:Y(I) =  INT (156.5 - SY * (P(1,I) - Y0)): NEXT
5030  FOR I = 0 TO N: HPLOT X(I) - 3,Y(I) TO X(I) + 3,Y(I): HPLOT X(I),Y(I) - 3 TO X(I),Y(I) + 3: NE
XT : RETURN
5040  HOME : VTAB 22: HTAB 9: PRINT "THESE ARE YOUR POINTS": GOSUB 640: RETURN
5059  REM

CALCULATE SPLINE PARAMETERS

5060  HOME : VTAB 22: HTAB 10: PRINT "SPLINE BEING FITTED": VTAB 1
5069  REM

PARAMETER A1

5070  FOR I = 0 TO N:A1(I) = P(1,I): NEXT
5079  REM

PARAMETER H

5080  FOR I = 0 TO N - 1:D(I) = P(0,I + 1) - P(0,I): NEXT
5085 D(N) = D(N - 1):D(N + 1) = D(0)
5089  REM

PARAMETER V

5090 B(0) = 2 * D(0): FOR I = 1 TO N - 1:B(I) = 2 * (D(I) + D(I - 1)): NEXT :B(N) = 2 * D(N - 1)
5099  REM

PARAMETER W

6000 C(0) = 3 * (A1(1) - A1(0)): FOR I = 1 TO N - 1:C(I) = (D(I - 1) * (A1(I + 1) - A1(I))) / D(I)
6010 C(I) = C(I) + ((D(I) / D(I - 1)) * (A1(I) - A1(I - 1)))
6020 C(I) = 3 * C(I): NEXT :C(N) = 3 * (A1(N) - A1(N - 1))
6029  REM

PARAMETER V'

6030 V(0) = B(0):V(1) = B(1) - (D(1) * D(N + 1) / V(0))
6040  FOR I = 2 TO N:V(I) = B(I) - ((D(I) * D(I - 2)) / V(I - 1)): NEXT
6049  REM

PARAMETER W'

6050 W(0) = C(0): FOR I = 1 TO N:W(I) = C(I) - (D(I) * W(I - 1) / V(I - 1)): NEXT
6059  REM

PARAMETER B

6060 B(N) = W(N) / V(N): FOR I = (N - 1) TO 1 STEP  - 1:B(I) = (W(I) - (D(I - 1) * B(I + 1))) / V(I)
: NEXT
6070 B(0) = (W(0) - (D(N + 1) * B(1))) / V(0)
6079  REM

PARAMETER C

6080 C(0) = 0:C(N) = 0: FOR I = 1 TO N - 1:C(I) = (A1(I + 1) - A1(I)) / (D(I) ↑ 2):C(I) = 3 * C(I):C
(I) = C(I) - (((2 * B(I)) + B(I + 1)) / D(I)): NEXT
6089  REM

PARAMETER D

6090  FOR I = 0 TO N - 1:D(I) = (C(I + 1) - C(I)) / (3 * D(I)): NEXT : RETURN
6130  REM

PLOT THE CURVE

6140  HPLOT X(0),Y(0): FOR I = 0 TO N - 1: FOR J = X(I) TO X(I + 1):TEMP = X0 + (J - 5) / SX
6150 TEMP = TEMP - P(0,I)
6160 TEMP = A1(I) + B(I) * TEMP + C(I) * TEMP ↑ 2 + D(I) * TEMP ↑ 3
6170 TEMP = 156 - SY * (TEMP - Y0) + .5: IF TEMP > 170 OR TEMP < 1 THEN  GOTO 6190
6180  HPLOT  TO J,TEMP
6190  NEXT : NEXT : RETURN
9999  REM

MAIN PROGRAM

10000  HGR : GOSUB 50000: TEXT : GOSUB 25: GOSUB 45000:S1Y = 145: GOSUB 40000: POKE  - 16304,0: HCOL
OR= 3
10010  GOSUB 5010: GOSUB 360: GOSUB 5040:
10020  GOSUB 5060: GOSUB 6140: GOSUB 25: GOSUB 50000: TEXT : END
39999  REM

SCALING FOR DRAWING

40000  FOR I = 0 TO N:A(I) = P(0,I): NEXT : GOSUB 190:SX = 260 /  ABS (A(N) - A(0)):X0 = A(0)
40010  FOR I = 0 TO N:A(I) = P(1,I): NEXT : GOSUB 190:SY = S1Y /  ABS (A(N) - A(0)):Y0 = A(0): RETUR
N
44999 REM

DATA  INPUT

45000  HOME : VTAB 5: HTAB 4: PRINT "YOU NOW HAVE TO ENTER YOUR DATA": PRINT : HTAB 5: PRINT "YOU FI
RST ENTER THE X-VALUES"
45010  HTAB 9: PRINT "AND THEN THE Y-VALUES ": PRINT : PRINT : PRINT : HTAB 11: PRINT "TO FINISH TYP
E '/'": GOSUB 25:N = 0
45020  HOME : VTAB 5: PRINT "WHAT IS X-VALUE OF POINT ";N + 1: PRINT "PRESS <RETURN> AFTER TYPING YO
UR NUMBERS": INPUT N$: IF  ASC (N$) = 47 THEN  GOTO 45070
45030  IF  ASC (N$) < 48 OR  ASC (N$) > 57 THEN  GOTO 45020
45040 P(0,N) =  VAL (N$)
45050  PRINT : PRINT "WHAT IS Y-VALUE  ": INPUT N$: IF  ASC (N$) < 44 OR  ASC (N$) > 57 OR  ASC (N$)
 = 46 OR  ASC (N$) = 47 THEN  GOTO 45050
45060 P(1,N) =  VAL (N$):N = N + 1: GOTO 45020
45070 N = N - 1:F(3) = 7:F(2) = 2: FOR J = 0 TO (1 + .4343 *  LOG (N))
45080  HOME : PRINT "READING  X-VALUE  Y-VALUE": PRINT : PRINT : PRINT : FOR I = 0 TO 9: IF I + 10 *
 J > N THEN  GOTO 45100
45090  PRINT 1 + I + 10 * J;:F(1) = P(0,I + 10 * J): GOSUB 40: PRINT  SPC( 6 -  LEN ( STR$ (1 + I +
10 * J)))F$;:F(1) = P(1,I + 10 * J): GOSUB 40: PRINT  SPC( 2)F$: NEXT
45100  PRINT : PRINT "IS DATA OKAY?  Y OR N?": GET T$: IF  ASC (T$) <  > 89 AND  ASC (T$) <  > 78 TH
EN 45100
45104  IF  ASC (T$) = 89 AND I + 10 * J > N THEN  GOTO 45125
45105  IF  ASC (T$) = 78 THEN  GOTO 45110
45110  PRINT "ENTER NUMBER OF WRONG SET ": INPUT X: HOME : PRINT "NOW ENTER X-VALUE FOR POINT ";X: I
NPUT P(0,X - 1)
45120  PRINT "& NOW THE Y-VALUE ": INPUT P(1,X - 1): GOTO 45080
45123  NEXT J
45125  HOME
45130  HOME : VTAB 5: HTAB 8: PRINT "CALCULATIONS BEING MADE": RETURN
49999  REM

INTRODUCTION

50000  HOME : VTAB 7: POKE 32,7: PRINT :A$ = "=======================": INVERSE : PRINT A$:A1$ = "=
                    =": PRINT A1$:X =  VAL (A$)
50010  PRINT A1$: PRINT A1$: PRINT "= CUBIC SPLINE FITTING =": PRINT A1$
50020  PRINT A1$: PRINT A1$: PRINT A$: NORMAL : POKE 32,0
50025  FOR I = 26 TO 35: READ F$(I): NEXT I: RESTORE : RETURN
```



'There's something at the door selling computers'

MAN MACHINE AND MOUTH



by Louis Belcourt

FROM PORT & CHIPS TO VOICE

A Microcomputer that can talk!

There are many methods to attach a peripheral such as a voice system, a terminal or a disk drive to a computer and the best method is confusing as there are so many ways that such links can be accomplished. Today, the way computers communicate with one another or to a specific device is undergoing a revolution of unprecedented proportions. We are just seeing the beginning of this revolution as the decade opens. Communication using optic fibres, light or laser beam are methods that are going through extreme development. It happens though, that physical interfaces are about the only level at which there is some universal agreement. The two common methods used are: a serial interface and a parallel interface.

SERIAL INTERFACE.

A serial interface in its minimal form uses only three wires to connect pheripheral device to the computer – one wire is for data transmited by the device to the computer, another is for data received by the device, and the third line is a ground connection. Information sent back and forth is usually coded into a special format known as ASCII (The American Standard Code for Information Interchange), which is a code that requires eight bits to represent any of the 128 characters plus other information. Data can also be received or transmited in EBCDIC (Extended Binary–Coded–Decimal Interchange Code) and there are other formats.

To transmit the information representing each character (in ASCII) over a single wire, special circuits must be used to convert the eight parallel bits into a stream of eight bits and vice versa. These circuits operate at various speeds and in several different modes. There are two basic operation modes: Synchronous and Asynchronous . The synchronous mode requires that information be transmitted at a specific time along with specific timing pulses and both the transmitting and receiving units must "lock" their timing signals together. Using synchronous mode allows data to be sent at rates of 56,000 bits per second and sometimes higher. Asynchronous systems, on the other hand, do not require both units to lock into each other. Instead, each time data are to be sent, the unit sending data interupts the other unit and sends the information. To ensure data are received properly a format was developed and it consists of a start bit, five to eight data bits, one or two stop bits and an even or odd parity bit. A parity bit indicates whether the sum of all one bits in the character is odd or even. Typical communication rates during asynchronous operation are 110, 300, 500, 1200 and 9600 bits per second commonly known as baud rates.

Most manufacturers agree on the RS-232-C interface (serial Port) between data terminal equipment and other peripherals or computers as standard. An RS-232-C interface allows a -5V to -15V level to represent a logic 1 and a +5V to +15V level for a logic 0, although +/-12V levels are commonly used. Another form of interface is the TTY (Teletypewriter) 20mA current loop where an approximate 20mA current flow indicates a logic 1 while absence of a current in the loop indicates a logic 0.

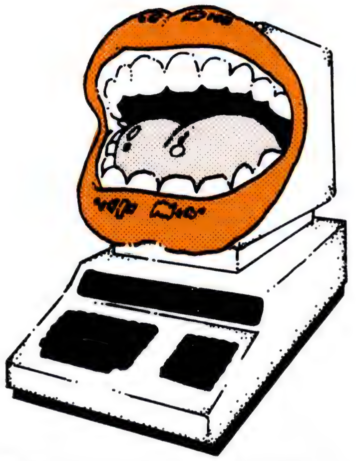
The heart of any serial interface is the circuit that will do the serial or serial-to-parallel conversion. These circuits, contained in a single IC almost as complex as the microprocessor they interface to, can accept parallel data words on one set of inputs and deliver a serial pulse train on another output. The other half of the circuit accepts a serial pulse train, removes the start, stop, and parity bits and delivers a parallel data word, There are many tvpes of such IC on the market and they are known as UART (universal asynchronous receiver/ transmitter), USRT (universal synchronous receiver/transmitter) and USART (universal synchronous/ asynchronous receiver/transmitter).

PARALLEL INTERFACE.

The other method of transmitting data back and forth is via a parallel interface. This method uses at least eight wires for the byte or character to be transferred and often two or more control lines to signal the computer or pheripheral that data are ready to be sent or have been received. Since eight bits are transmitted at one time, and often in a synchronous manner, parallel informations transferred are High-speed operations with rates of 1 million bits per second possible.

What is a port? Yes it is a strong sweet dark-red (occasionally brown or even white) fortified wine of Portugal; Port being shortened from the word "Oporto", city of Portugal, whence shipped. But it can also be described as a gateway, or an opening for entrance, loading, etc; and although both descriptions will mix well with CHIPS (jargon for Integrated Circuit Devices) the latter is more appropriately suited to computer logics. My definition of port is a hardware channel for a computer to transmit and receive data from an external peripheral device. There are two important points to keep in mind here namely Channel and Data.

Most computers are equipped with more than one channel or port and there must be a way to select which



port the computer will receive (Input) or send data (Output). TheZ80 Microprocessor which is the CPU I will use here to interface to a voice system, can directly communicate with UP to 256 ports. Special instructions provided in the Z80 are just intended to transfer information from the Processor to the interface device namely OUT (N), A or IN A' (N) where N is the port number (0 to 255) e.g the OUT (120), A instruction transfers data from the Z80 accumulator (register A) to port number 120.

Various methods can be employed to design a port system and I think the choice of parts depends more on the cost rather than on the complexity of circuit design. Fig. 1a shows a simple circuit to decode address of port number 255 (Binary equivalent being 11111111). This circuit uses an 8-input NAND gate (NOT AND). The output will go low (logic 0) only when all the 8 input lines are high (logic 1).

If one or more of the address A0 to A7 in Fig 1a was inverted to a logic 0, then any of the 256 port address in the Z80 could be selected. e.g if A6 is at logic 0 and connected to an inverter, as in Fig 1b, all the input to the 8-input NAND gate will still be at logic 1 and the new address 10111111 will be decoded. Fig 1c shows a method of decoding an 8 bit address using a switch to determine which address is to be at level 0 or 1. Only 4 address lines are selected although the circuit can be duplicated on address A0 to A3.

Note that the Z80 has an 8 bit data Bus but has a 16 bit address Bus. Data is transferred 8 bits at a time (D0 to D7) and whilst address lines A0 to A15 are used to address memory, only lines A0 to A7 are used in a port system. This is why the maximum number of ports available on the Z80 is 256 (0 to 255); 255 binary 11111111 being the maximum number permissible using 8 bits.

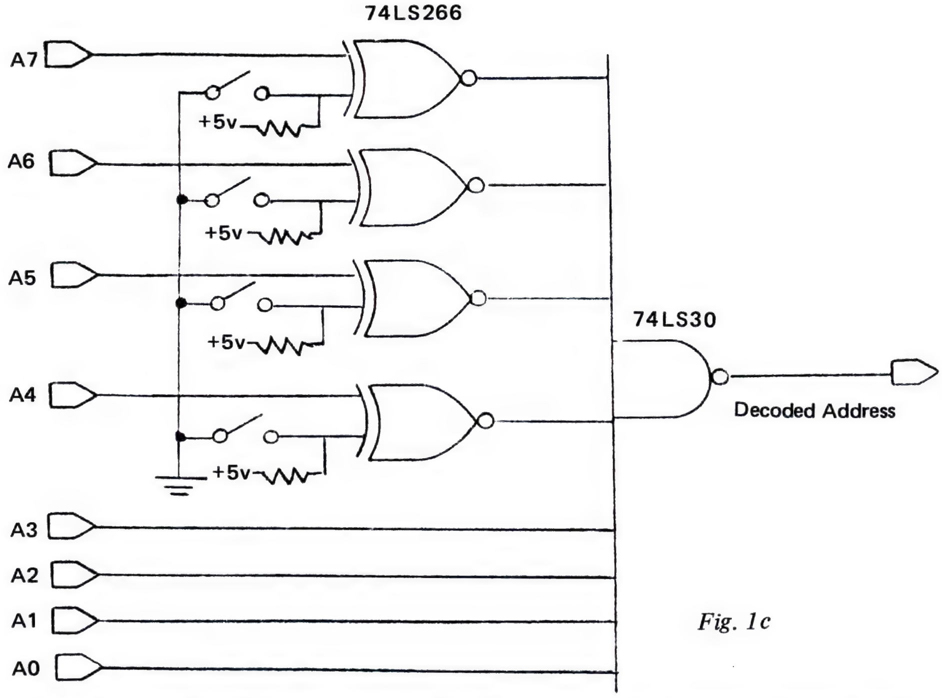


Fig. 1c

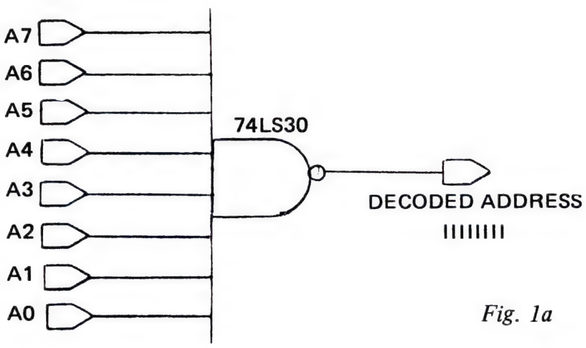


Fig. 1a

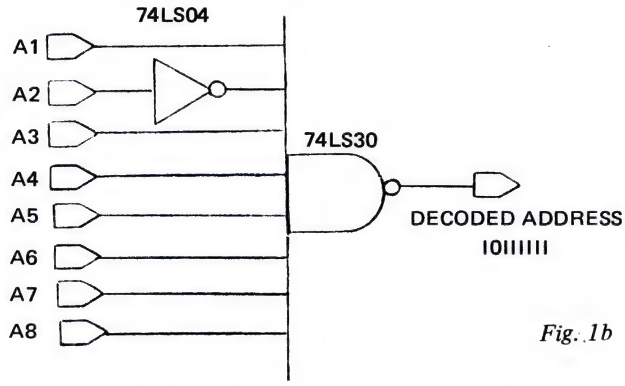


Fig. 1b

CONTROL SIGNALS

After decoding the address A0 to A7, there must be a way to tell whether the address refers to memory or an I/0 port. The Z80 uses four control signals to keep data flowing at the right time and in the right direction. They are:

MREQ which is the memory request signal. It indicates that the address present on the address bus is valid and a read or write operation can then be performed on the memory.

I/O REQ is the input/output request. It indicates that I/O address present on bits 0 to 7 of the address bus is valid and an I/O read or write operation can then be carried out.

RD is the memory-read signal. It indicates that the Z80 is ready to read the contents of the data bus into its accumulator.

WR is the memory-write signal. It indicates that the data bus holds valid data, ready to be written into a specific device.

Monitoring these four lines is all the information necessary to control any I/O port. An I/O operation (e.g to or from a port) is designated by the I/O REQ control signal being at logic 0. The RD and WR signals control the direction that data flows along the data bus. Fig 2 shows how these signals can be combined to form one I/O read and I/O write strobe signal.

Note: The bar on top of the signal names indicates an active low state (logic 0).

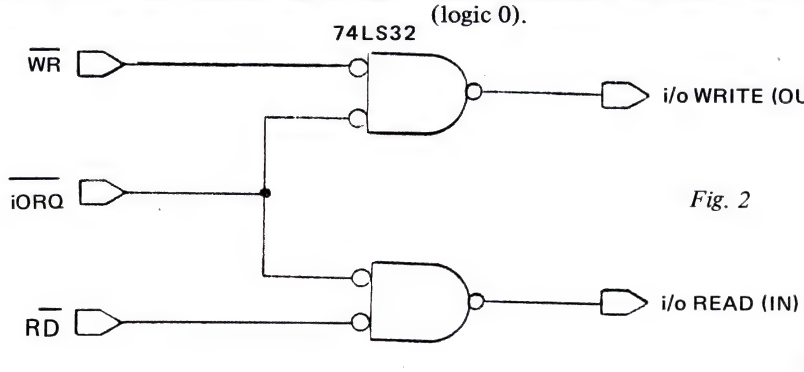


Fig. 2

When the instruction OUT (N), A is executed, the contents of the accumulator (which holds the data to be transferred) is placed on the data bus. This data is actually valid during only a few hundred nanoseconds and additional circuitry must be provided to latch or hold the contents of the data bus. This type of circuitry is often referred to as an "8-bit latched parallel output port". The latch also serves another purpose. It is not advisable to connect external devices directly to the data bus due to the possibility of interference or bus-loading problems. Hence the latch circuitry is also used as a buffered gate to allow signals from the data bus to be sent to the external devices. During an input operation the process used for output is reversed.

As for serial interface, instead of the variety of IC required for decoding, signal control, output latching and input buffering, these circuits are all combined in one LSI (large scale integrated) chip. The Intel or National Semiconductor 8255 PPI Programmable Peripheral Interface) IC is one example which contains three eight-bit ports which may be used for input or output along with quite a few other circuits.

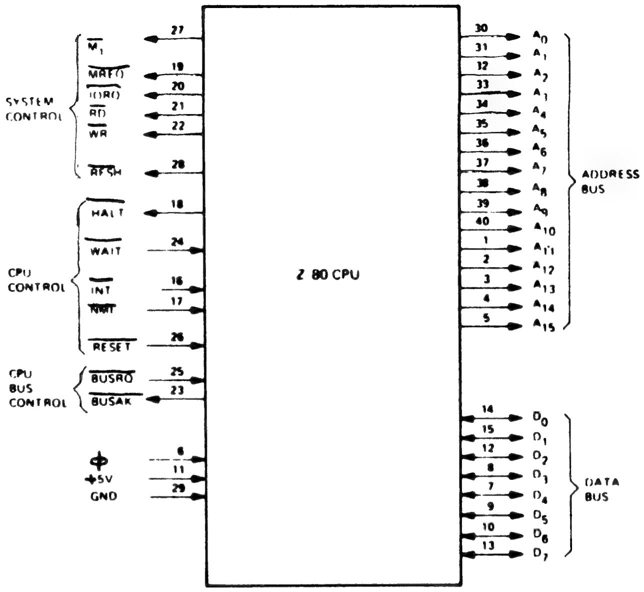


Fig. 3a

VOICE INTERFACE

The voice system that I will use to interface to a Z80 microprocessor or as a matter of fact to any computer, uses the DIGITALKER kit from National Semiconductor Corp. Although this kit can operate on its own, its full capabilities can only be achieved when connected to a computer and controlled by a program. The kit contains 137 separate words, 2 tones and 5 different silence durations. The silence durations are provided to improve the quality of any phrase by inserting the durations between words. Fig 3a shows the Z80 CPU P in configuration, Fig 3b shows the recommended schematic diagram which consists of the Speech Processor Chip (SPC), speech ROM, filter, amplifier and speaker.

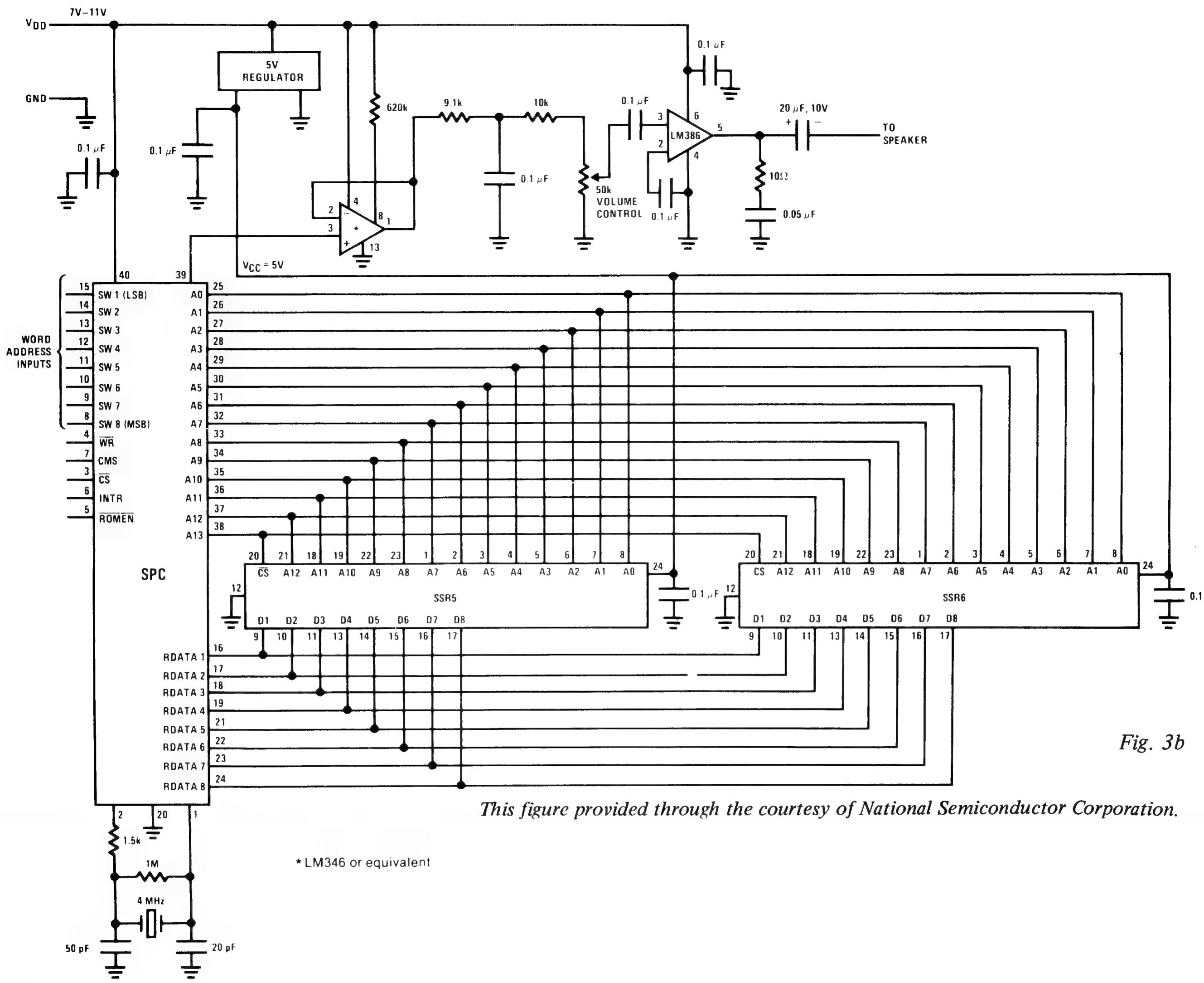


Fig. 3b

This figure provided through the courtesy of National Semiconductor Corporation.

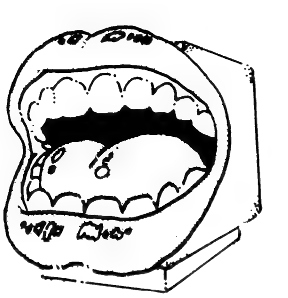
IT'S alm

ost

HERE

Australian Business Computer, the new magazine for business decision-makers, is on the news stands this month. Don't miss it. Fill out our reply-paid card for a home delivered copy of the first issue, at a special price for APC readers. Avoid troubling your local newsagent who will be overwhelmed with purchasers of ABC. No stamp required.

Australian
BUSINESS
COMPUTER



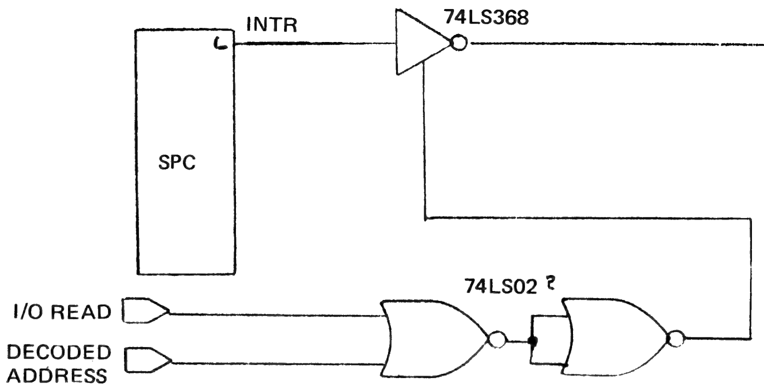


Fig. 4

SW1-8 is an 8-bit parallel input to the SPC that allows all of the 137 words contained in the ROm to be generated by using a unique code. Connecting the SPC to a selected port system allows a microprocessor to send any of the codes in 8 parallel bits making it possible to output single words or words concatenated into phrases or even sentences.

WR is the signal that latches the SW1-8 data into a register. This latching of data will simplify the design of an inteface in the sense that the information on the data bus is held or latched by the SPC and the 8-input data can be directly connected to the Z80 data bus. On the rising edge of WR, the SPC starts execution of the command specified by CMS. (CMS is a command select line. Logic 0 resets the interrupt and starts the speech sequence while logic 1 resets the interrupt only). If a command to start generating a new word is issued while a word is being outputed, the new word will overlap the previous word. What this means is there must be a way to check for a completion of a word before issuing the next one. Fortunately such a signal is provided on the SPC (INTR) which goes high at the completion of a word being issued. If the INTR line is not used, and say that, four words are generated one after another, it is most probable that only the last word will be spoken. If the same word is generated with a loop of calculated timing in between, the outputed word will sound like stuttering: e.g. if the word "please" (code 120) is generated four times without checking the INTR line but adding a small program loop in between each word, PL–PL–PL–PLEASE will be produced. Perhaps words can be merged to expand the available vocabulary; e.g. the word "forget" is not available in the ROM (not yet anyway) but the word "forward" (code 51) and the word "get" (code 54) can be produced. Could the word "for-get" be generated?

Could the word "for-get" be generated?

CS (Chip select) is the line to enable the SPC (Logic 0).

ROMEN is used to drive a switching transistor that can be linked to the voltage line of the ROM in a low power application.

CIRCUIT DESCRIPTION

As mentioned before, SW1-8 (Pins 8 to 15 of the SPC) can be directly connected to data bus D0 to D7.

$\overline{CS}$ enables the SPC when low. This line can be connected to the decoded port address. e.g in Fig. 1c the decoded address will go low only when A0 to A7 is valid for the selected port number, thus the SPC will be active only when the correct port number is used.

$\overline{ROMEN}$ need not be connected. The power consumption of the two ROMs is less than 200mA. This line could be very useful in a low power stand alone unit or when more than two ROMs are used.

$\overline{WR}$ can be connected to the I/O write signal (refer Fig 2). This line will latch the data on the data bus in the register contained within the SPC to be processed and eventually transformed as spoken words.

INTR is the line to check the completion of a spoken word. Fig 4 shows a simple circuit to check through the use of the IN instruction whether the INTR line is high or low. This line can be connected to any of the 8 data lines D0 to D7 but must be enabled only during the I/O read so as to avoid confilct with data going to the SPC. Combining the I/O read and the selected port signal allows the IN instruction to use the same port number as the OUT instruction.

This basically completes the description of the SPC data and control signal lines and how they can be interfaced to a port system. Note that most computers such as the TRS80, APPLE II, SYSTEM 80, or any S100 bus system already have most of the control, address and data lines on their edge connector. This makes the DIGITALKER kit a very easy adaptable voice system and at a cost well within the reach ot the microcomputer enthusiast.

IN CONCLUSION.

As more ROMs become available from National Semiconductor Corp., this project seems to be the thing of the future. Under control of a computer, words can be spoken by issuing just one instruction instead of using complicated software. Application enhanced by speech output are limitless. At home or in business speech will eventually be part of every computer.

Fig 5 shows a simple circuit for adding additional ROMs to the DIGITALKER kit.

The following program when used in any disk operating sytem on a TRS80 or SYSTEM 80 will speak the word "READY" as well as display the word on the screen.

```
                 ORG 41ACH
3E7F  START  LD   A,7FH
D380         OUT  (128),A  REM
                  is a port no.
C9           RET
             END START
```

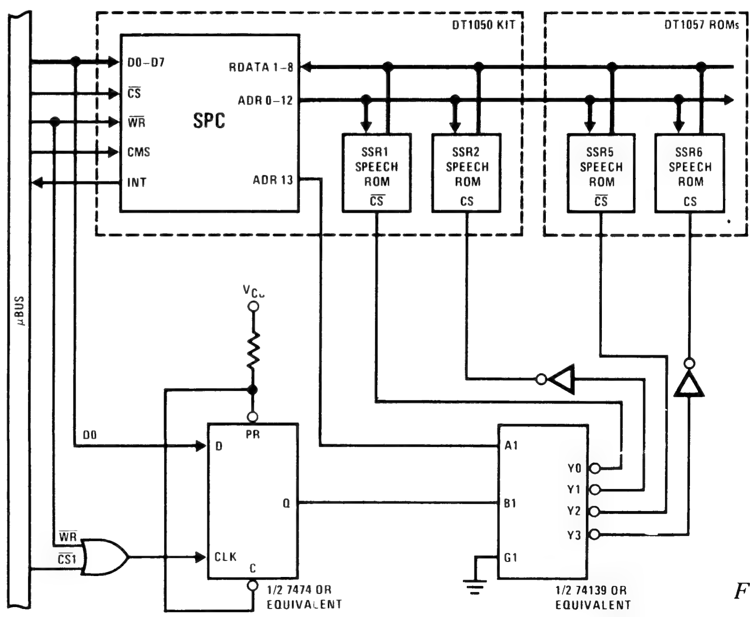


Fig. 5

Integration of DT1057 ROMs and DT1050 Kit

PRICE BREAKTHROUGH

A DAISY WHEEL PRINTER

$995

MICRO-80 has converted the revolutionary new Olivetti Praxis 30 portable electronic DAISY WHEEL typewriter to operate as a correspondence quality printer. Now you can have the best of both worlds. The best auto-correcting portable electronic typewriter available AND a reliable word processing printer for your microcomputer.

Designed for private and light commercial use, the MICRO-80 conversion provides the PRAXIS with an industry standard Centronics printer port which will operate with most microcomputers including the TRS-80TM, the SYSTEM 80TM the SORCERER and the APPLE. Average print speed is 120 words per minute. An RS232C serial to Centronics converter is also available from MICRO-80 to enable the PRAXIS or any other Centronics compatible printer to operate from an RS232C serial port.

MICRO-80 is an A Grade Olivetti distributor and has been producing printer conversions for Olivetti daisy wheel typewriters for 12 months. All typewriters sold by us carry our Australia-wide 90 day parts and labour warranty. If your need is for a larger, heavier duty typewriter/printer, contact us for details of the ET121 or ET221 conversions.

The demand for this great value printer is sure to be high. Use the coupon below to secure your order today.

DEALER ENQUIRIES WELCOME

TM: TRS-80 is a trademark of the Tandy Corporation
SYSTEM-80 is a trademark of Dick Smith Electronics Pty. Ltd.

MICRO-80 433 MORPHETT STREET, ADELAIDE, S.A. 5000
TELEPHONE: (08) 211 7244

Post now to:
MICRO-80 PTY. LTD., P.O. BOX 213, GOODWOOD, S.A. 5034

Please send me:

		TICK
A PRAXIS 30 Typewriter converted to operate as a Centronics compatible printer	$995	☐
An RS 232C to Centronics converter	$140	☐
More details of the above products		☐

I enclose my cheque/money order of $. . . . .
NOTE: Prices above are F.O.B. Adelaide
Please add $10.00 for road freight anywhere in Australia

Please charge to my Bankcard
Bankcard No. ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐
Expiry date: . . . . / . . . .

Name: . . . .
Address: . . . . Post Code: . . . .

The same result can be achieved by converting the Hex code into decimal numbers and poking them at address 41ACH using a Basic program.

In the field of computer technology, there is a tremendous potential for use of speech output. "Enter a number, please" could be a command issued by a computer to an operator without visual capabilities. "Error Try Again" or "Invalid profit margin" are spoken words that can even be useful to any person using a computer.

Many handicapped persons could benefit from such a marvel of today's technology. A compacted battery operated microprocessor system incorporating a voice synthesizer can provide voice for individuals striken with neurological or physical disorders which impair the human voice mechanism.

"Hello Darling I love you" could be words spoken by a person never able to speak before. Isn't is a wonderful world.

WORD LIST WITH DECIMAL & HEX ADDRESSES

WORD	DEC ADDR	(HEX)
THIS IS DIGITALKER	0	00
ONE	1	01
TWO	2	02
THREE	3	03
FOUR	4	04
FIVE	5	05
SIX	6	06
SEVEN	7	07
EIGHT	8	08
NINE	9	09
TEN	10	0A
ELEVEN	11	0B
TWELVE	12	0C
THIRTEEN	13	0D
FOURTEEN	14	0E
FIFTEEN	15	0F
SIXTEEN	16	10
SEVENTEEN	17	11
EIGHTEEN	18	12
NINETEEN	19	13
TWENTY	20	14
THIRTY	21	15
FORTY	22	16
FIFTY	23	17
SIXTY	24	18
SEVENTY	25	19
EIGHTY	26	1A
NINETY	27	1B
HUNDRED	28	1C
THOUSAND	29	1D
MILLION	30	1E
ZERO	31	1F
A	32	20
B	33	21
C	34	22
D	35	23
E	36	24
F	37	25
G	38	26
H	39	27
I	40	28
J	41	29
K	42	2A
L	43	2B
M	44	2C
N	45	2D
O	46	2E
P	47	2F
Q	48	30
R	49	31
S	50	32
T	51	33

WORD	DEC ADDR	(HEX)
U	52	34
V	53	35
W	54	36
X	55	37
Y	56	38
Z	57	39
AGAIN	58	3A
AMPERE	59	3B
AND	60	3C
AT	61	3D
CANCEL	62	3E
CASE	63	3F
CENT	64	40
400HERTZ TONE	65	41
80HERTZ TONE	66	42
20MS SILENCE	67	43
40MS SILENCE	68	44
80MS SILENCE	69	45
160MS SILENCE	70	46
320MS SILENCE	71	47
CENTI	72	48
CHECK	73	49
COMMA	74	4A
CONTROL	75	4B
DANGER	76	4C
DEGREE	77	4D
DOLLAR	78	4E
DOWN	79	4F
EQUAL	80	50
ERROR	81	51
FEET	82	52
FLOW	83	53
FUEL	84	54
GALLON	85	55
GO	86	56
GRAM	87	57
GREAT	88	58
GREATER	89	59
HAVE	90	5A
HIGH	91	5B
HIGHER	92	5C
HOUR	93	5D
IN	94	5E
INCHES	95	5F
IS	96	60
IT	97	61
KILO	98	62
LEFT	99	63
LESS	100	64
LESSER	101	65
LIMIT	102	66
LOW	103	67

WORD	DEC ADDR	(HEX)	WORD	DEC ADDR	(HEX)
LOWER	104	68	PULSES	124	7C
MARK	105	69	RATE	125	7D
METER	106	6A	RE	126	7E
MILE	107	6B	READY	127	7F
MILLI	108	6C	RIGHT	128	80
MINUS	109	6D	SS*	129	81
MINUTE	110	6E	SECOND	130	82
NEAR	111	6F	SET	131	83
NUMBER	112	70	SPACE	132	84
OF	113	71	SPEED	133	85
OFF	114	72	STAR	134	86
ON	115	73	START	135	87
OUT	116	74	STOP	136	88
OVER	117	75	THAN	137	89
PARENTHESIS	118	76	THE	138	8A
PERCENT	119	77	TIME	139	8B
PLEASE	120	78	TRY	140	8C
PLUS	121	79	UP	141	8D
POINT	122	7A	VOLT	142	8E
POUND	123	7B	WEIGHT**	143	8F

*"SS" MAKES ANY SINGULAR WORD PLURAL
**ADDRESS 143 IS THE LAST LEGAL ADDRESS IN THIS PARTICULAR WORD LIST. EXCEEDING ADDRESS 143 WILL PRODUCE PIECES OF UNINTELLIGIBLE INVALID SPEECH DATA.

VOX II WORD LIST WITH DECIMAL & HEX ADDRESSES

WORD	DEC ADDR	(HEX)	WORD	DEC ADDR	(HEX)
ABORT	0	00	DOOR	31	1F
ADD	1	01	EAST	32	20
ADJUST	2	02	"ED"	33	21
ALARM	3	03	EMERGENCY	34	22
ALERT	4	04	ENTER	35	23
ALL	5	05	ENTRY	36	24
ASK	6	06	"ER"	37	25
ASSISTANCE	7	07	"EHT"	38	26
ATTENTION	8	08	EVACUATE	39	27
BLUE	9	09	EXIT	40	28
BRAKE	10	0A	FAIL	41	29
BUTTON	11	0B	FAILURE	42	2A
BUY	12	0C	FAHRENHEIT	43	2B
CALL	13	0D	FAST	44	2C
CALLED	14	0E	FASTER	45	2D
CAUTION	15	0F	FIFTH	46	2E
CELSIUS	16	10	FIRE	47	2F
CENTIGRADE	17	11	FIRST	48	30
CHANGE	18	12	FLOOR	49	31
CIRCUIT	19	13	FOURTH	50	32
CIGAR	20	14	FORWARD	51	33
CLOSE	21	15	FROM	52	34
COLD	22	16	GAS	53	35
COMPLETE	23	17	GET	54	36
CONTINUE	24	18	GOING	55	37
COPY	25	19	GREEN	56	38
CORRECT	26	1A	HALE	57	39
CREASE	27	1B	HEAT	58	3A
"DE"	28	1C	HELLO	59	3B
DEPOSIT	29	1D	HELP	60	3C
DIAL	30	1E	HURTS	61	3D

WORD	DEC ADDR	(HEX)
HOLD	62	3E
HOT	63	3F
IN	64	40
INCORRECT	65	41
INTRUDER	66	42
KEY	67	43
LEVEL	68	44
LIGHT	69	45
LOAD	70	46
LOCK	71	47
LONGER	72	48
MORE	73	49
MOVE	74	4A
NEXT	75	4B
NO	76	4C
NORMAL	77	4D
NORTH	78	4E
NOT	79	4F
NOTICE	80	50
OPEN	81	51
OPERATOR	82	52
OR	83	53
PASS	84	54
PER	85	55
POWER	86	56
PRESS	87	57
PRESSURE	88	58
PROCESS	89	59
PULL	90	5A
PUSH	91	5B
PUT	92	5C
QUARTER	93	5D
RANGE	94	5E
REACHED	95	5F
RECEIVE	96	60
RECORD	97	61
REVERSE	98	62
RED	99	63

WORD	DEC ADDR	(HEX)
REPAIR	100	64
REPEAT	101	65
REPLACE	102	66
ROOM	103	67
SAFE	104	68
SECOND	105	69
SECURE	106	6A
SELECT	107	6B
SEND	108	6C
SERVICE	109	6D
SIDE	110	6E
SLOW	111	6F
SLOWER	112	70
SMOKE	113	71
SOUTH	114	72
STATION	115	73
SWITCH	116	74
SYSTEM	117	75
TEMPERATURE	118	76
TEST	119	77
"TH"	120	78
THANK	121	79
THIRD	122	7A
THIS	123	7B
TURN	124	7C
UNDER	125	7D
USE	126	7E
WAITING	127	7F
WARNING	128	80
WAS	129	81
WATER	130	82
WEST	131	83
WIND	132	84
WINDOW	133	85
YELLOW	134	86
YES	135	87
ZONE	136	88

COMPUTER GAMES

BRIDGE

by David Levy

Contract Bridge is one of the most interesting and skilful of card games, ranking alongside poker in its complexity. Many computer programmers are also bridge enthusiasts, so I expect that many of my readers will, at some time or another, have considered the possibility of writing their own bridge program. Let me warn you from the outset — this is a most daunting task. I would expect a competent programmer to take three times as long to write a bridge program as to write a fairly respectable chess program, and the size of the bridge program would be much larger — with less than 32 kbytes you might as well forget it. But since most of you who own or have access to computers *will* have at least 32k at your disposal, writing a bridge program is a task that can be undertaken by anyone who is prepared to devote a lot of time and effort.

In writing about computer bridge I shall attempt only to outline some simple principles which will enable the reader to write a working program. The game is sufficiently rich in ideas that stronger bridge players will be able to extrapolate from my article and include a number of more advanced concepts in their programs. Anyone who writes a 'simple' bridge program based on these articles will be able to enjoy an undemanding game without the need to find three other (human) players.

How to play bridge

I do not wish to go into a detailed description of the rules of the game, but some of my readers may not know how to play, so some explanation is essential. This will also enable those of you who play other bidding games to learn the principles of programming the bidding phase, which can be carried over to other games. The principles of playing the cards might also be useful in programming other games which are based on taking tricks. So don't give up if you are not attracted to the idea of programming bridge — what you learn here may help you in other games.

Contract bridge is played by four players, who form two partnerships. A normal deck of 52 cards is used, and at the start of a hand each player is dealt 13 cards. The players start by bidding for the right to play the hand, and whichever side makes the highest bid then tries to make the number of tricks indicated by that bid. If the

Illustration Ingram Pinn

partnership is successful in making the desired contract it scores points according to the size of the bid, and the number of extra tricks (overtricks) made by the partnership. If the contract is not made, the partnership which was playing to make the contract loses penalty points, the number of penalty points depending upon the number of tricks by which the contract failed, and whether or not the defending partnership decided to 'double' the contract (doubling, as one might expect, doubles the number of points gained or lost on a hand, and also affects bonus points which can be scored in certain situations).

The player who deals the cards opens the bidding. He may say 'pass' or 'no-bid' if he doesn't wish to make a positive bid at this stage, or he may make a bid of the form 1 Club, or 2 No Trumps — the number part of the bid indicates the number of tricks that must be made if this bid is the final contract (number of tricks = number bid + 6); the suit part of the bid indicates what will be the trump suit if this bid is the final contract.

After the first player has made his bid or passed, it becomes the turn of the player sitting to his left. This player may also pass; or he may make a higher bid; or he may double the opponent's previous bid. In order to make a higher bid he must indicate a greater number of tricks, or he must bid a higher ranking suit (or no-trumps) than the previous bid. The ranking of the suits goes (from lowest to highest) Clubs, Diamonds, Hearts, Spades and then comes No-Trumps. So one Heart is higher than one Diamond; one No-Trump is higher than one of any suit; and two Clubs is higher than any bid at the one level.

The bidding proceeds in this way, moving round the table in a clockwise direction and coming to an end only when three successive players pass. Even if a player doubles or redoubles an opponent's bid, there must then be three successive passes before the bidding is at an end. Once the bidding is over, the players who won the bidding (ie made the final bid) are obliged to play the contract, or to be more precise, whichever of them first bid the suit (or No-Trumps) of the final contract — he is the one who must play the cards for his partnership. He is referred to as 'Declarer' and his partner as 'Dummy',

The player on the declarer's left leads any card that he chooses and, at this point, dummy places all of his cards on the table, face up so that everyone can see them. From now on dummy has nothing to do until the hand is over. His partner, declarer, must play the cards for both of them.

The rules of play are very similar to those of Whist and many other trick-taking games. The player who wins one trick leads the first card to the next trick, and players must follow suit if possible, or if this is not possible they may trump a card if they possess any trumps. (In a No-Trump contract this is not applicable.) Cards rank in the usual order, from Ace, King, Queen and Jack down to 4, 3 and 2.

The bidding phase

The point of the bidding phase is to try to reach the optimal contract, partly by conveying information to your partner about the strength and 'shape' of your hand. In order to be able to determine what contract you and your partner should be playing, it is important for you to know something about each other's hand. This is accomplished by the bidding, but because every bid must be higher than the previous bid, a partnership does not have a completely free license to pass information back and forth during the bidding, as this would lead them into an impossibly high contract. So the most important thing to do during the bidding is to try to reach the ideal contract by conveying the maximum information about your hand in the most economical manner. Let us examine the bidding of a hand of bridge to see how information is conveyed.

		Spades:	K 8 7 4 2			
		Hearts:	2			
		Diamonds:	A J 9 5 3			
		Clubs:	K 10			
Spades:	10		N		Spades:	9 5
Hearts:	Q J 10 7 6				Hearts:	K 9 4
Diamonds:	Q 8 2	W		E	Diamonds:	K 10 7
Clubs:	Q J 4 3		S		Clubs:	A 9 8 6 5
		Spades:	A Q J 6 3			
		Hearts:	A 8 5 3			
		Diamonds:	6 4			
		Clubs:	7 2			

For the sake of convenience we usually refer to the four hands by the four points of the compass: North, South, East and West. We shall assume that West was the dealer, and that the bidding goes like this: (players' thought processes in brackets).

West: Pass (I have a weak hand);
North: One diamond (I have a stronger-than-average hand with two good suits. I shall bid the lower ranking suit first to give my partner a chance of bidding hearts at the one level);
East: Pass (I also have a hand that is no better than average, and since my partner is weak we will not have enough combined strength to make any contract);
South: One spade (I have two biddable suits, but I have more spades than hearts so I shall bid spades first);
West: Pass;
North: Two spades (My partner has at leat four spades in his hand so we at least nine spades out of 13 between us. Obviously spades will be a good suit for us to play a contract in);
East: Pass;
South: Three hearts (I must show my partner that I have another biddable suit);
West: Pass;
North: Three spades (My first spade bid indicated only that I had reasonable spade support for my partner. Now I should tell him that I have more than minimal spade support and that I do not have enough strong cards in the unbid suits to make a no-trump contract possible);
East: Pass;
South: Four spades (My partner has at least four spades and probably holds the king of spades. He also has four or five diamonds so he does not have many clubs and hearts. I have the Ace of hearts so we are unlikely to lose more than one heart trick, and I only have two clubs so we cannot lose more than two club tricks before I can trump any further clubs that are led. So we ought to be able to avoid losing any more than three tricks, and four spades seems quite possible);
West: Pass;
North: Pass (Enough is enough);
East: Pass.

The above bidding and thought processes represent an over-simplification of what was going on in the minds of the players. But it does serve to explain the type of thought processes that one goes through when bidding in a simple fashion. I ought perhaps to mention at this stage that by reaching certain contracts a partnership may qualify for a 'game bonus' if the contract is made. These game contracts are: 3 No-Trumps; 4 Hearts or 4 Spades; 5 Clubs or 5 Diamonds. Making a lesser contract allows you to score the game bonus later on if you can make another contract that counts, together with the earlier contract, for enough points to make a game. I will not go into the scoring system in this article, but you should study an elementary book on bridge before writing your program, so that the scoring will be correct.

In order to make the bidding phase easier and to ensure that information is conveyed economically, various bidding systems have been invented. In a bidding system, each bid has a fairly precise defined meaning, and by correctly interpreting a bid, a player will understand more about his partner's hand. One useful tool employed in many bidding systems is what are known as 'high card points'. This points method usually counts 4 points for holding an Ace, 3 for a king, 2 for a queen, 1 for a jack or singleton (a suit with only one card, other than an Ace), 4 for a void (a suit with no cards), 1 for each card after the first five in a suit. Using this point count method, various rules of thumb have been developed, including:

a) Do not open the bidding with fewer than 12 points;
b) If you hold 12-15 points you should open one of your best suit.
c) If you hold 16-18 points you should open one No-Trump.
d) In order to make a three No-Trump contract the combined hands should have not less than 24 points, preferably 25 or more.

The above rules can all be broken, under the correct circumstances and, in fact, the same bid can mean many different things in the same situation, depending on which system of bidding

the partnership is employing. The most important thing to remember about bidding is that bridge is a partnership game, and you should be trying to help your partner during the bidding by making meaningful bids that he will understand. There is no point in making a brilliant bid on one bidding system if your partner is using a different system — he will not understand what you mean and before you know what is happening you and your partner will have overbid, and found yourselves in an impossible contract.

How to program a bidding system

Before writing your program, decide what bidding system will be used in the program and make a long list of what the various bids can mean in different circumstances. Whenever the program must make a bid it determines the circumstances and makes the appropriate bid. Whenever the program must interpret a bid made by its partner, it determines the circumstances under which the partner's bid was made, and then looks at the list of bids to see what the particular bid should mean in those circumstances. These two processes, the making of the correct bid and the interpreting of the partner's bid, can each be aided by keeping a number of important variables and updating them in the light of new information transmitted or received. The following variables might usefully be employed when deciding what bid to make or when interpreting a bid made by one's partner:

Max Clubs (what is the maximum number of clubs that have been shown so far by the player who is bidding this hand);
Min Clubs (the minimum number of clubs shown by the bidding);
Max Diamonds
Min Diamonds
Max Hearts
Min Hearts
Max Spades
Min Spades

By storing values for all the above variables, the program can build up an idea of the way in which the suits are distributed in his partner's hand, or he can keep track of the extent to which he has described the distribution of the suits in his own hand. In addition to knowing how long a suit might be, it is also very useful to have some indication as to how strong a particular hand might be.

This can be accomplished using two variables called Max Points and Min Points, which indicate the known limits of strength of a hand as indicated by the number of high card points in the hand. For example, if a partnership is using a bidding system in which 13 points is the minimum number for making an opening bid, a player who makes an opening bid is known to have at least 13 points so his Min Points is initially adjusted from 0 (the default value when the hand is dealt) to 13.

Adjusting the distribution variables is not a particularly difficult matter. At the start of a hand the four Max variables are set at numbers which may be deduced from the holding of the hand under scrutiny. For example, if the computer is making the first bid for West in the above hand, it sets Max Spades for North, East and South at 12, since it has one spade and knows that no other player may therefore have more than 12. Similarly, Max Hearts is set at 8, Max Diamonds is 10 and Max Clubs is 9. The minimum values of the suit variables are all set at 0 since no bids have been made and therefore nothing is known about the distribution in each of the hands other than the program's 'own' hand.

If we follow through the bidding of the above hand again, assuming that the computer is playing South, we can see how easy it is to adjust the distribution variables for the other hands. (Here I shall make certain assumptions concerning the bidding systems employed by the N-S pair and the E-W pair.)

West: (West's Max Points is set to 1, as he would open the bidding on 12 or more). West's Min Points remains at 0.
North: One diamond (North's Min Points is set at 12, Max Points is set at 15, since with 16-18 points North would have opened one No-Trump, and with 19 or more he would have opened two of a suit.) Also, Min Diamonds is set at 4, the minimum number needed to bid, and Max Diamonds is set at 7, since with 8 he would have opened higher.)
East: Pass (East's Max Points = 11, Min Points = 0)
South: One spade (South, the program, has indicated that he holds at least 7 points, otherwise he would have passed. So Min Points = 7, Max Points = 11, otherwise he would have made a stronger bid to indicate that he, too, held an opening hand. Min Spades = 4 and Max Spades = 6, since with seven or more spades, South would have made a stronger bid than one spade.)
West: Pass
North: Two spades (Min Spades = 3, Max Spades = 5, since with six or more spades North would be able to bid higher in spades, and would have opened in spades rather than diamonds. Also Max Hearts = 4 and Max Clubs = 4, by subtraction from 13.)
East: Pass
South: Three hearts (Min Hearts = 4, Max Hearts = 6 and, by subtracting from 13, we find that Max Diamonds = Max Clubs = 4. Note that neither clubs nor diamonds can be longer than a four card suit, as this would have required South to bid the suit before now.)
West: Pass
North: Three spades (Min Spades = 4)
East: Pass
South: Four spades (Min Spades = 5)

This example is not intended to indicate exactly how the variables should be adjusted, nor is it intended to be complete in the summary of information conveyed by each bid. The sole *raison d'etre* for the example is to show the reader the type of information that can be gleaned from a bid, and how this information may be used to update some of the more useful variables. When you have decided on the bidding system that will be employed in your program, the method for updating each of the variables will suggest itself.

Special conventions in bidding

There are a number of special bidding conventions, each of which may be used in a particular situation. Often these conventions take the form of a question and an answer. For example, the Blackwood convention is a method of asking your partner how many aces he holds, and how many kings. This information is particularly useful if your partnership is hoping to make a small slam (12 tricks) or a grand slam (13 tricks). The asking bid in Blackwood is 4 No-Trumps, and the replies are:

5 clubs, when holding no aces (sometimes this reply is given when holding all four aces);
5 diamonds, when holding one ace;
5 hearts, means two aces;
5 spades, means three aces.

In order to ask how many kings your partner has you simply bid five no-trumps, and he bids the number of kings at the six level (6 clubs is 0 or maybe 4), 6 diamonds is 1, etc.

When the Blackwood convention is employed, the program can update variables such as: Number of Aces, Number of Kings, and the tri-state variables Ace of Clubs, Ace of Diamonds, etc, which can indicate yes, no or don't know, depending on what may be deduced from the bidding. For example, if you hold two aces and find that your partner holds the other two, you know which aces he holds and so you can set the values of the tri-state variables (Ace of Clubs, etc) accordingly. This detailed use of variables can be most helpful when making a slam decision.

Another popular convention is known as Stayman, and consists of a two club asking bid after your partner has bid one no-trump. The asking bid enquires whether partner has at least four cards in either hearts or spades (or both), in which case he should respond by bidding the appropriate suit (or the better suit if he holds at least four cards in each of the two suits). If the program asks this question of its partner, it can use the reply to update the variables Min Spades, Max Spades, Min Hearts and Max Hearts, according to the reply bid.

Deciding what to bid-a simple algorithm

When faced with the decision of what bid to make, a number of complex factors enter the thought processes of a good bridge player. Here we are discussing the problems of writing a relatively simple bridge program, and so we must try to employ a relatively simple bidding algorithm. I have devised such an algorithm, which lacks the subtlety of an advanced bridge player, but which ought to provide the computer with the ability to make bids that are reasonably intelligible and reasonably sensible. The algorithm applies to any bidding system, so you may choose any system that you like, preferably from a good book on bidding. One word

of advice — try to use a 'natural' bidding system (one in which the bids tend to reflect the obvious features of the hand) rather than an 'artificial' system (in which most of the bids form an apparently obscure code).

Most books on bidding will offer advice on how many high card points are needed to make contracts at various levels. In 'Bridge for Beginners' (by Victor Mollo and Nico Gardener), for example, we find that a useful guideline is:

22-25 points are needed in the combined hands to make any contract

26 points are needed to make 3 No-Trumps, 4 Hearts or 4 Spades, or 5 Clubs or 5 Diamonds

34 points are needed to make a slam (12 or 13 tricks)

These guidelines are extremely useful, inasmuch as they can set an upper limit on the program's bidding. In our earlier example, once the program knows that it and its partner (playing North-South) hold less than 34 points, which is when the second bid is made (South's one Spade), it is immediately obvious that a slam is not a real possibility, so the maximum contract is a game contract and the highest possible bid is 5 Diamonds).

The manner in which the algorithm operates is simplicity itself. The program first asks the question 'can I bid again without exceeding the safe limit?', where the safe limit is defined by the above guidelines. If the answer to this question is 'yes', the program simply examines every one of its legal bids, determines what would be meant by each of these bids, and then performs some sort of matching exercise to produce a numerical score that represents the accuracy with which each bid describes the hand (bearing in mind what has already been bid). In a situation where the program is responding to an asking bid (eg, Blackwood or Stayman) there is no problem — the program simply gives the correct answer to the asking bid. But in the general case the program must evaluate each bid and then choose the bid with the highest score, or, if two or more bids have a similarly high score, the program selects the lower bid so that it can convey information in an economic manner.

How exactly this matching procedure is programmed will depend entirely on the type of bidding system you employ in your program, but a few hints may be useful for setting you on the right track. Firstly, we should consider a situation in which the program ought to make an asking bid. This might happen when it has discovered that it and its partner have 34 points or more between the two hands. The program may wish to know how many aces and kings are in its partner's hand (unless it has all the aces and kings itself), in which case it bids 4 No-Trumps. If the answer indicates that its partner has all the missing aces, the program can then ask how many kings are in its partner's hand. It will then find out whether the partnership is missing any of the important top cards and make its decision as to whether it can afford to bid 7 (for a Grand Slam) or only 6 (a Small Slam). Asking bids and their responses are as easy for a computer program as for a human player.

In a more general situation, the program must decide the extent to which a bid conveys information that has not already been conveyed. One way to do this is to count the number of variables which can be updated after making a particular bid. If a bid provides information which gives useful information about three of the variables, then the bid is, in some sense, more useful to the program's partner than a bid which gives useful information about only two variables.

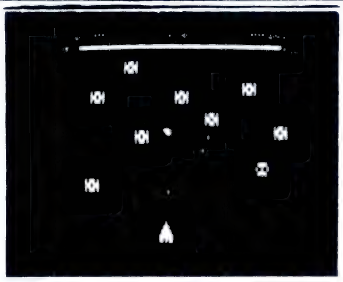
One final point, which is important to implement because of the necessity of playing a contract in the best suit (or in No-Trumps if that is better than a suit contract): throughout the bidding the program should keep some kind of measure for each suit and for No-Trumps. This measure should indicate the desirability of playing a contract in that suit. At the start of the hand, when the cards are dealt, the measures might simply be the number of high card points in each of the suits (excluding the points for singletons and voids). For No-Trumps the measure should be zero. When the program's partner makes a bid, the number of high card points for the suit bid should be increased by (say) 8 for the first time that partner bids the suit, 4 for the second time and 2 for the third time. If the program's partner bids all of the suits in which the program does not have adequate control (either an ace, or a king and one other card, a queen and two other cards or a jack and three other cards), then the number of points assigned to No-Trumps can be adjusted to some high value (say 15). Each time that the partner bids another suit, which he has not yet bid, this score is increased by 2. The program then has a relatively easy measure of whether each suit is worthwhile, and whether No-Trumps is a possibility. Then, as the level of the bidding gets nearer and nearer to the guideline limits, the program can easily make a decision about the final contract. It is then only important to avoid making a bid which is so high that partner can no longer make a safe bid (ie, a bid within the guideline limits) in a suit which is deemed to be acceptable.

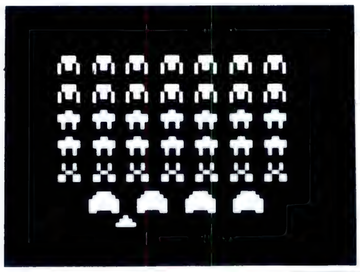
Next month, I shall write about playing the hand once the bidding is over. In the meantime, I suggest that you find a good book on bidding and select an easily programmable system.

FREE CATALOGUE

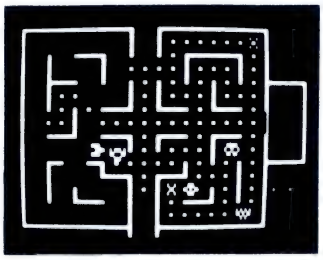
Enter the Compshop Australia
ARCADE CENTER

COSMIC FIGHTER
For TRS80 Model 1/3 & SYSTEM 80 16K tape . . .
$19.50



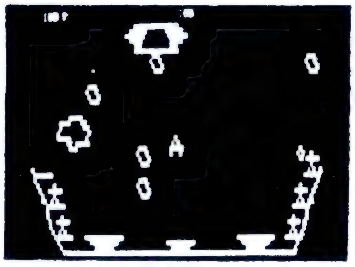


SPACE INTRUDERS
For TRS80 Model 1/3 & SYSTEM 80 16K tape . . .
$19.50

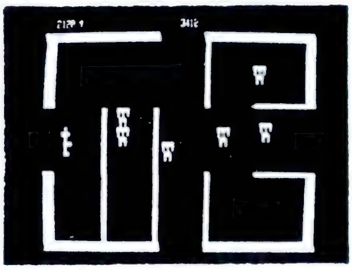


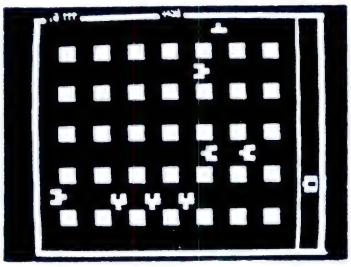
GOBBLEMAN
For TRS80 Model 1/3 & SYSTEM 80 16K tape . . .
$19.50

METEOR MISSION 2
For TRS80 Model 1/3 & SYSTEM 80 16K tape . . .
$19.50



ROBOT ATTACK
For TRS80 Model 1/3 & SYSTEM 80 16K tape . . .
$19.50

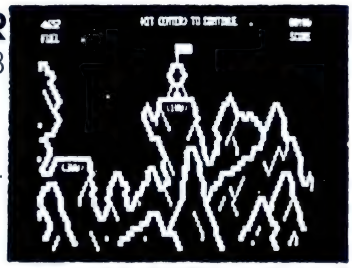




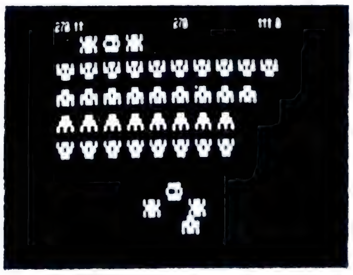
ATTACK FORCE!
For TRS80 Model 1/3 & SYSTEM 80 16K tape . . .
$19.50

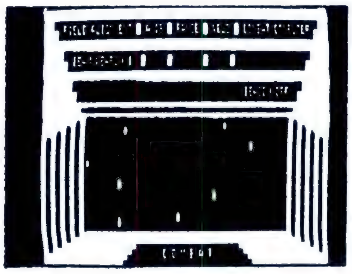
SUPER NOVA
For TRS80 Model 1/3 & SYSTEM 80 16K tape . . .
$19.50

LUNAR LANDER
For TRS80 Mod 1/3 & SYSTEM 80 16K tape . . . **$19.50**
ATARI 24K tape . . . **$24.50** (Requires joysticks)



GALAXY INVASION
For TRS80 Model 1/3 & SYSTEM 80 16K tape . . .
$19.50





STAR FIGHTER
For TRS80 Model 1/3 & SYSTEM 80 16K tape . . .
$39.50

GUARANTEE Compshop guarantees to replace any tapes immediately if you are unable to load them into your computer.

Compshop Australia, Office 4, 75 Palmerston Crescent, South Melbourne, Victoria. Phone (03) 690 5336. APC 2/82

I have .... micro

☐ Please send me your software catalogue. I enclose a stamped self-addressed envelope.

☐ Please send me .... I enclose a cheque/money order $..........................

Plus $1 for postage. Please debit my bankcard A/C for $..........................

A/C No. .... Exp ....

Name ....

Address ....

Postcode .... Signature ....

COMPUCOLOR COLUMN

by Ralphe Neill of the Victorian Users' Group

By now the exciting news must have started filtering through to many of you that the Compucolor II is no longer an orphan-child!

Software and hardware support for the Compucolor and Intecolor microcomputers is now assured!

For those of you who haven't heard the details, here's a brief run-down of what's been happening.

Intelligent Systems Corp. decided last year to stop making the CCII partly because of the new FCC regulations on RFI and partly because they were losing money on the deal.

ISC had finally discovered what we knew all along – the CCII was grossly under-priced when compared with other micros fitted with the options necessary to bring them up to the CCII's capabilities.

All was gloom! The new ISC product – the Intecolor 3651 – was essentially software-compatible with the CCII, but we didn't have a great deal of faith in the ISC's ability to produce the goods.

Now all that has changed with a new company called Intelligent Computer Systems – ICS for short (confusing isn't it?).

It seems that ICS will effectively be ISC's main marketing arm and will be handling all Compucolor and Intecolor software and hardware.

The software will include that from ISC itself as well as the top-quality products coming in from users – things like COMP-U-WRITER and a machine language data-base that does searches in seconds rather than minutes!

There's not likely to be any hardware from ISC for the CCII – but ICS will be handling products developed by users.

LOCAL DISTRIBUTION

All of the above would be rather pointless if there wasn't also some support in Australia, but a national distributor and a Victorian agent had already been appointed at the time of writing. The national distributor is:–

Color Computer Systems Pty Ltd.,
58 Valley Road,
Hornsby, NSW 2077

The man to speak to there is Tony Sforcina who has already brought thousands-of-dollars-worth of software into the country and Victorian users can get it at:–

Panatronics Pty Ltd.,
691 Whitehorse Road,
Mont Albert, Vic. 3127.

The General Manager of Panatronics is Neil Brandie – well-known to Victorian users through his enthusiastic help and support.

By the time this appears, other agents will probably have been appointed and I'll give you the details in the next column.

THE COMPUCOLOR III

No, that's not a misprint – there could well be a Compucolor III appearing on the market soon.

It is to be built in Italy under licence from ISC. It will support 8 inch drives and it seems that it will be similar in performance to the Intecolor 3651.

Nothing more than that is known at the moment, but I'll keep you informed.

If nothing else, it's a further indication that our favourite micro is far from dead!

QUICK TIPS

The tip in the last Compucolor Column about using (ESCAPE) (E) after hitting (ESCAPE) (W) by mistake has already saved quite a few programs – so here's a few more tips that, if not quite as useful as the last one, will at least save a few key-strokes.

Did you know that, when in FCS, you can use keys other than RETURN to initiate a command?

Try ERASE PAGE when listing a DIR to a cluttered screen. You can also use A7 ON and A7 OFF.

And hitting the down-arrow instead of RETURN after entering DIR will list only the directory header – useful if you just want to identify a disk that has a lot of directory entries.

"AUSSI" GRAPHICS

This column's program listing gives a quite spectacular display. It comes from Mr D. Niven of the NSW Users' Group.

The DATA statements do look a little daunting but the results will be well worth the effort.

And, as the display it initially produces is a cleaned-up version of the map of Australia from an early Australian ColorCue, you could save considerable time by using those earlier DATA statements if you've saved them.

THE AUSTRALIAN SOURCE

The Australian Source will be operating by the time you read this. Initially, it's in Melbourne only but will be operating in Sydney within three months and then in other capital cities at further three-month intervals.

You may have noticed from the advertisements that the Compucolor II is not one of the "approved" systems.

But don't worry too much – what it means is that there will be no programs available specifically designed for the CCII as there will be for the "approved" micros.

All of the other facilities of the Australian Source will, of course, be available and there's nothing to stop us building up our own libraries with world access.

I'll put my Australian Source account number in the next Column so you can leave messages for me – in the meantime, as ever, you can write to me care of APC.

```
100 REM  "AUSSI" BY D. NIVEN
120 PLOT 6,2,15,12 : INPUT "ENTER COLOUR-CODE (1-255) FOR PLOT 6
,X STATEMENT: ";YY
130 PLOT 6,YY,12,3,64,0 : I=180 : HI=127 : LO=63 : FOR J=1 TO I
140 READ X0,Y,X1 : PLOT 2,250,X0,Y,X1,255 : NEXT J
150 DATA 20,20,24,19,21,26,18,22,26,19,23,27,20,24,28
160 DATA 20,25,30,19,26,31,19,27,31,19,28,34,18,29,35
170 DATA 18,29,35,17,30,37,17,31,39,17,32,39,16,33,39
180 DATA 16,34,40,15,35,41,15,36,44,14,37,45,14,38,46
190 DATA 13,39,49,13,40,51,12,41,112,12,42,113,11,43,113
200 DATA 58,40,111,62,39,111,63,38,72,73,39,111,64,38,71
210 DATA 72,38,110,64,37,71,72,37,110,64,36,71,74,36,109
220 DATA 65,35,70,74,35,108,66,34,69,74,34,108,66,33,68
230 DATA 73,33,107,67,32,67,73,32,107,73,31,73,72,30,72
240 DATA 72,29,72,74,31,106,75,30,106,75,29,76,77,29,105
250 DATA 75,28,75,78,28,105,79,27,104,79,26,104,79,25,103
260 DATA 79,24,103,79,23,103,80,22,102,80,21,102,81,21,102
270 DATA 81,20,100,82,19,90,92,19,100,83,18,89,92,18,98
280 DATA 84,17,87,92,17,95,93,16,94,85,16,86,11,44,114
290 DATA 10,45,114,10,46,114,9,47,114,9,48,114,8,49,115
300 DATA 8,50,115,8,51,9,11,51,116,7,52,8,11,52,115,10,53,115
310 DATA 10,54,115,9,55,114,9,56,114,8,57,114,8,58,114
320 DATA 8,59,115,7,60,115,7,61,115,8,62,115,8,63,115
330 DATA 7,64,114,7,65,114,7,66,114,7,67,7,9,67,113,10,68,113
340 DATA 10,69,112,11,70,112,12,71,111,12,72,111,13,73,110
350 DATA 13,74,110,14,75,109,14,76,109,16,77,110,18,78,110
360 DATA 19,79,106,20,80,106,21,81,105,22,82,105,24,83,104
370 DATA 25,84,104,26,85,104,26,86,103,27,87,102,27,88,102
380 DATA 28,89,101,28,90,100,29,91,99,29,92,98,29,93,97
390 DATA 29,94,30,32,94,97,30,95,30,33,95,96,33,96,95
400 DATA 32,97,95,32,98,95,35,99,76,80,99,94,35,100,74
410 DATA 82,100,94,35,101,72,82,101,94,36,102,71,83,102,93
420 DATA 36,103,71,83,103,93,36,104,69,83,104,93,36,105,46
430 DATA 50,105,68,83,105,93,37,106,42,49,106,67,83,106,93
440 DATA 39,107,40,48,107,65,84,107,93,48,108,64,83,108,92
450 DATA 49,109,64,83,109,91,49,110,65,83,110,89,49,111,66
460 DATA 83,111,90,50,112,66,83,112,87,50,113,66,84,113,87
470 DATA 51,114,67,83,114,87,55,115,67,83,115,87,57,116,60
480 DATA 83,116,86,84,117,86,84,118,85,84,119,85
490 DATA 85,11,87,93,11,94,85,10,94,86,9,94,87,8,93
500 DATA 87,7,92,88,6,91,88,5,91,89,4,90
510 PLOT 30,2,246,0
520 FOR Q=0 TO HI : PLOT Q,HI : NEXT
530 FOR Q=HI TO 0 STEP -1 : PLOT Q,HI : NEXT
540 FOR Q=0 TO HI : PLOT 246,0,Q,HI,250,0,Q,HI : NEXT
550 PLOT 250,0 : FOR Q=0 TO HI : PLOT Q,HI : NEXT
560 PLOT 246,0 : FOR Q=LO TO 0 STEP -1
570 PLOT Q,HI,HI-Q,HI : NEXT
580 PLOT 250,0 : FOR Q=LO TO 0 STEP -1
590 PLOT Q,HI,HI-Q,HI : NEXT
600 FOR Q=0 TO HI
610 PLOT 246,0,Q,HI,HI-Q,HI,250,0,Q,HI,HI-Q,HI : NEXT
620 PLOT 246,0 : FOR Q=0 TO LO : PLOT Q,LO,HI-Q,LO : NEXT
630 PLOT 250,0 : FOR Q=0 TO LO : PLOT Q,LO,HI-Q,LO : NEXT
640 PLOT 246,0 : FOR Q=0 TO HI : PLOT Q,HI : NEXT
650 FOR Q=0 TO LO : PLOT Q,LO,HI-Q,LO : NEXT
660 PLOT 250,0 : FOR Q=HI TO 0 STEP -1 : PLOT Q,HI : NEXT
670 PLOT 250,LO+1 : FOR Q=0 TO HI : PLOT Q,HI : NEXT
680 FOR Q=127 TO 0 STEP -1
690 PLOT 246,0,Q,LO,246,LO+1,HI-Q,HI,250,LO+1,Q,HI
700 PLOT 250,0,HI-Q,LO : NEXT
710 FOR Q=LO TO 0 STEP -1
720 PLOT 246,0,Q,HI,HI-Q,HI,250,0,LO-Q,HI,HI-LO+Q,HI : NEXT
730 PLOT 255 : GOTO 510
```

Relocatable Assembly Language Programs

by Jeff Richards

The ability to write assembly language routines that can run at any memory address without recompilation can be extremely valuable in some circumstances. The popular microprocessors have some serious limitations in this regard, but with a few simple tricks most of these limitations can be overcome.

In order to be able to write relocatable programs we must be able to use relative addressing. This is an instruction being executed. Relative addressing must be available for jumps, subroutine calls and data references, but most popular microprocessors do not have all these instruction types. The 8080 has no relative-addressing instructions, while the Z80, 6800 and 6502 have relative jumps only. (The Signetics 2650 has a complete set of relative addressing instructions, although the range is limited to +63 and -64 bytes).

The examples presented here are written for the Z80 microprocessor, but they apply to any processor that has a common data and subroutine return stack that is in main memory. In order to see how they work it is essential to understand the details of the subroutine call and return instructions and the way they use the stack.

The stack is simply an area of main memory that is set aside for temporary storage of values. It differs from other areas of main memory in one important aspect - there is maintained, within the CPU, a register which points to the next available storage spot in this area of memory. This means that instructions to access data in this area can be very simple, as the CPU knows the precise memory location we are concerned with. The only limitation we have to accept is that data must be retrieved in the reverse sequence to that in which it was stored - hence the term "stack".

Instructions to access the stack include PUSH, which places the data contained in one of the CPU registers into the stack, and POP which retrieves the top value from the stack and places it into a nominated register. These single-byte instructions automatically update the stack pointer, so the programmer does not have to worry about the precise memory location at which the data is stored.

Similarly, he does not have to worry about whether the stack grows up or down, and whether the stack pointer is updated before or after the instruction execution. All these details are looked after by the CPU, so using the stack for temporary storage is quick and easy.

The routines to be considered here use the stack in a similar way, but in this case the data to be stored will be an address. This will be achieved by using the features of the subroutine instruction (CALL) and the return-from-subroutine instruction (RETurn).

The CALL instruction pushes the address of the next sequential instruction onto the stack, and then jumps to the address specified in the instruction. The RETurn instruction takes this data on the top of the stack and jumps to the address it specifies. That is, the stack is used to store the subroutine return address, and its "last in - first out" structure means it is well suited to this task. The fact that these instructions happen to use the stack can generally be ignored by the programmer, but there are some situations in which he must be aware of what is happening.

The major impact of the fact that the CALL and RETurn instructions use the stack is that any subroutine must leave the stack the way it found it. If a subroutine does more PUSHes than it does POPs, or vice versa, then when the RETurn is executed the return address will not be at the top of the stack, with very strange results. Providing this requirement is realised, the values on the stack can be used by the programmer in any way he wishes.

Describing the CALL and RETurn instructions in terms of the way they manipulate the stack provides a clue to some of the things that can be done. Put simply, the CALL does a PUSH of the program counter before updating it with the address of the subroutine, while a RETurn pops the stack into the program counter. Note that to "execute a jump" really means to modify the program counter in some way, so both CALL and RETurn are varieties of jumps.

An example of how the stack can be manipulated in this way can be found in a subroutine that has to use data from memory. Such a subroutine would be a message printing routine, where the message is a string of characters stored in memory and terminated with a special character. The usual procedure

for doing this is set out in Listing 1.

For a slight increase in the complexity of the subroutine, the first step in the calling routine can be dispensed with altogether. This is likely to be worthwhile, as the subroutine will be called from many different places in the program, so simplifying the calling procedure will save a significant number of program bytes. But it also has the advantage of removing an absolute memory reference – an essential step towards making the program.

The CALL will push the address of the next sequential byte onto the stack. If this happens to be the address of the message, then we don't have to tell the subroutine where to find the message. The drawback is that the subroutine has to calculate the address to return to, but this is done "automatically" as a consequence of incrementing the memory pointer through the message (revised version in listing 2). (Notice that (HL) means the contents of the memory location pointed to by the current value of the HL register pair.)

The CALL MESSAGE instruction pushes the address DATA onto the stack and then jumps to address MESSAGE for the next instruction. This instruction gets the item at the top of the stack (which happens to be the address of the message) and continues to print the string of characters at this address. Eventually, the terminator is detected, and the PUSH instruction puts the current value of the memory pointer (HL) onto the stack. This is the address which the RETurn pops off the stack and jumps to, which just happens to be the first byte after the end of the message, which is just what we want.

For an extra four bytes in the subroutine we have saved 3 bytes every time the subroutine is called. It is also possible to substitute the EX (SP), HL instruction for the POP and PUSH instructions, thus preserving the contents of HL.

However, removing the absolute memory reference is of much use if we still have an absolute subroutine call. Getting rid of this is a little more complex.

Obviously, what is needed is a procedure to put the return address onto the stack before executing a relative jump. The program counter can give us the return address, but there are no instructions that permit us to access the current value of the program counter. We could use the CALL instruction to push the return address onto the stack, but we are trying to get rid of the absolute addressing inherent in the CALL instruction.

The only way to solve the problem is to execute a call to a small routine that does not have to be relocated, and retrieve the return address for our relative jump. If this routine is small enough it can be tucked away in a corner of memory that is not being used. It should be possible to find 7 bytes somewhere in memory where this routine can be stored.

What does the routine have to do? All we are trying to achieve is a relative jump with a suitable return address on the top of the stack. We can get the program counter onto the stack with a CALL, so if we can return from the subroutine without modifying the top of the stack we can execute the relative jump as if it was a relative call. Therefore, this routine has to push a copy of the return address onto the stack. However, when we return from the absolute subroutine we still have not executed the relative jump that we are using as a relative call, so the actual return address we need is two bytes after the byte after the CALL! Fortunately, it's easier to program than it is to describe.
See Listing 3.

Before executing a relative call, the main program does an absolute call to SETUP. At this stage the address of the JR instruction is on the top of the stack. The POP/PUSH sequence gets this address into HL without disturbing the stack, and it is incremented twice. HL now contains the address of BACK, which is the address that the relative subroutine should return to.

The EX instruction puts this onto the top of the stack while retrieving the address of the JR instruction, and the PUSH then pushes this address onto the stack. Thus, the RETurn will come back to the JR instruction, and the instructions starting at SUBR will be executed with the address of BACK on the top of the stack. The RETurn at the end of SUBR will jump to the address on the top of the stack – BACK.

Of course, the RETurn in the relative subroutine can be conditional. As a major use of this technique is likely to be in converting existing programs to a relocatable form, it may be inconvenient to have register pair HL unavailable for passing parameters to the subroutine. In this case it would be possible to replace all occurrences of HL with IX or IY, if these registers are available. Otherwise HL could be restored after the call to SETUP, but any instructions inserted between the CALL and the JR must be matched by extra INC instructions within SETUP, or the address on the top of the stack will not be correct.

It would also be possible to locate SETUP at one of the RST addresses and call it using the RST instruction. This has the advantage that the relative call occupies the same three bytes as an absolute call does. Combining this with the fact that the relative subroutine uses the standard RETurn instruction means that modifying existing routines is quite feasible.

However, the real power of this routine becomes evident when it is combined with the above method of passing the address of data to the subroutine. Using these two techniques in conjunction provides us with a procedure that, with some careful programming, permits completely relocatable programs.

Note that although these procedures could be described as "fooling with the stack", they are in fact quite safe, and can be used in an interrupt driven environment. This is because the stack is only used in CALLS and RETurns, and the stack pointer never has to be adjusted. They have the advantage over other methods of relocating programs as the code never modifies itself.

There would be some circumstances where the variable length of the data makes this procedure impossible, or perhaps the program connot be made small enough to have subroutine calls no more than 128 bytes away from the subroutine. But for those circumstances where relocatability is critical, ways can be found around most of these problems. It would be quite easy, for instance to write a memory test program that moved itself up and down in memory to ensure that all of memory was adequately tested.

The limited addressing modes of the popular microprocessors, compared with the 2650 for instance, makes some programming techniques difficult. But at least the problem of relocatability is one that can be got around with a little bit of tricky programming.

LISTING 1.
Message Printing Subroutine.

```
          LD    HL,DATA
          CALL  MESSAGE
          . . .
MESSAGE:  LD    A,(HL)
          INC   HL
          CP    EOT
          RET   Z
          CALL  PRINT
          JR    MESSAGE
. . .
DATA:     DEFB  "This is the
                message", EOT
```

LISTING 2.
Same Routine without the Absolute Memory Reference.

```
          CALL  MESSAGE
DATA:     DEFB  "This is the
                message",EOT
          (Next Instruction)
. . .
MESSAGE:  POP   HL
M1:       LD    A,(HL)
          INC   HL
          CP    EOT
          JR    Z,FINISH
          CALL  PRINT
          JR    M1
FINISH:   PUSH  HL
          RET
```

LISTING 3.
Relative Subroutine Call.

```
SETUP:    POP   HL
          PUSH  HL
          INC   HL
          INC   HL
          EX    (SP),HL
          PUSH  HL
          RET
. . .
MAIN:     CALL  SETUP
          JR    SUBR
BACK:     (Next Instruction)
. . .
SUBR:     (More instructions)
          RET
```

The Binary Sort

by Robert Phillips

Here is a concise description of the Binary Sort concept, and a detailed implementation in Basic that should be easy to adapt to any micro or application.

Sometimes we have an array of data which we need to search in order to find the location of one particular element in it. This is more common with alphabetic data, but we may have to do it with either alpha or numeric data. The simplest way to find the item is to use a FOR-loop, checking each item individually until we find the one we are looking for. The average number of steps through the loop that must be made to find a given item is approximately half the length of the list. If the item is not on the list, then the program must execute as many steps through as there are items on the list. When the array is short, there is no problem. However as the array gets longer, this method becomes more and more inefficient. An array that has 500 elements in it will require an average of 250 steps through the loop to find an item. Such a search will take several seconds.

When the list is ordered (i.e. sorted into either ascending or descending order), there is a much more efficient way to search the list: the binary search. Basically stated, in a binary search you continually divide the list into two halves and then eliminate the half which cannot contain your item. (Because the list is always divided into two halves, this is called a binary search.) For example, if the item at the half-way point is larger than the item you are looking for, you know that your item [illegible] the second half [illegible] it from [illegible] the [illegible] he [illegible] til [illegible] ot [illegible]

After step 8 through the search, you have either found your item (and you may well have found it before step 8), or your search has failed. At any rate, it took you only 8 times through the loop to find your item, as opposed to the average of 128 (maximum: 255) that a straight search would require. The best part is that if you double the list, the binary search requires only one more step through the loop; double it again, and add just one more time through! Obviously, this is a wonderful tool.

Step No.	Eliminated this step	Total eliminated
1	128	128
2	64	192
3	32	224
4	16	240
5	8	248
6	4	252
7	2	254
8	1	255

Table 1.

Step	PT	IV	Find?	New IV	+ *or* −	New PT
1	8	8	no	4	+	12
2	12	4	no	2	−	10
3	10	2	no	1	+	11
4	11	1	YES!			

Table 2.

There are only two requirements for a binary search: 1) the list must be in order; and 2) the items on the list must be unique, (or, if not, it doesn't matter to you which of the duplicated items is located).

To do a binary search, we need two variables. One to point at where we are in the array, and one to keep cutting the search-field in half. In table 2, I [illegible]all them PT (for "pointer") and IV [illegible]or "interval"). IV will get cut in half [illegible]ch time through, until it gets down to [illegible] IV will be added to PT if we have to [illegible] further down the list; it will be sub[illegible]cted from PT if we have to come [illegible] higher on the list. To illustrate [illegible] assume an array of 15 [illegible] are searching for happens to be in position 11. Let's step through and see what happens to PT and IV at each step.

The logic to do this is not difficult. Let's say that our array is called L1$, and is an alpha array sorted into ascending (i.e. alphabetical) order. We have another variable TL ("total" – it is the same variable we would have used in a FOR-loop; FOR I=1 to TL) which tells us how many items are currently in the array. Finally, the item we are trying to find is stored in the variable SW$. The simple algorithm appears in figure 1.

If the array were sorted into descending order, the ">" and "<" symbols in statements 40 and 50 would be reversed. Notice that we use the INT function and round up. This is the equivalent to the CEILING function. Both things are necessary; if you don't round up, you won't be able to get to the end of the list, and non-integers will get clobbered during the division process.

As it happens, I do not like the redundancy of lines 40 and 50; I prefer to make them a little more efficient. I do it so that IV is always added to PT. Then with one compare, I find out if IV should be positive (so that the addition will add IV to PT) or negative (so that the addition will, in effect, subtract IV from PT). So, I prefer to have lines 40 and 50 as follows:

```
40 IF L1$ (PT)>SW$ THEN IV = -IV
50 PT = PT + IV
```

While this is certainly more "elegant", it also adds a problem. IV will quite often turn out negative, and that will really foul up what happens in statement 30. So, we have to change 30 to:

```
30 IV=INT ((ABS(IV))/2+.5).
```

```
10  PT=INT(TL/2+.5: IV=PT
20  IF  L1$(PT)=SW$  THEN  GOTO
    [you have found it!]
30  IV=INT(IV/2+.5)
40  IF  L1$(PT)>SW$  THEN  PT=PT–
    IV
50  IF  L1$(PT)<SW$  THEN  PT=PT+
    IV
60  GO TO 20
```

Figure 1

Now, having added the ABS function into line 30 to ensure that IV will always be positive, I am not sure that I have gained anything in efficiency. But, I think that it is more elegant, so I'll leave it!

If you try to run the program the way it is, you may have a problem: if the item that you are searching for is not on the list, you will get into an infinite loop and the only way out of the algorithm is to find the item. So, we have to check to see if IV has the value of 1. If it does we cannot cut in half any more; we cannot search any more. We need to test IV's absolute value, and I put it right after the compare, calling it line 25.

25 IF ABS(IV)=1 THEN GOTO (the search has failed)

If everything in the world were perfect, that would be the algorithm. However, since consistently rounding IV up for the reasons pointed out above, we may actually, at some times, exceed the bounds of the array, raising the error condition. There are several different ways to handle the problem; I believe the easiest is to take the value of IV away from PT and continue on from there. Since I don't know at this point if IV is negative or positive, I simply change its sign and add it to PT in line 55.

55 IF PT>TL OR PT>1
THEN IV = -IV:PT=PT + IV

(if you really don't like to have IV go negative and then to have to use ABS, you can use the original version of lines 40 and 50, and then use two statements here in place of 55.

IF PT<1 THEN PT=PT + IV
and IF PT > TL THEN PT=PT-IV)

My version of the binary sort algorithm is shown in figure 2.

```
10  PT=INT(TL/2+.5): IV=PT

20  IF L1$(PT)=SW$ THEN GOTO [found it! PT
    is the number of the item]

25  IF ABS (IV)=1 THEN GOTO [the search
    has apparently failed]

30  IV=(INT((ABS(IV))/2+.5)

40  IF L1$(PT)>SW$ THEN IV=-IV

50  PT=PT+IV

55  IF PT>TL OR PT<1 THEN IV=-IV: PT=PT+IV

60  GOTO 20
```

Figure 2

There is, unfortunately, still one more potential problem. If the number of items in the array (TL) is exactly a power of 2 (16, 32, 64, 128, etc), the search will not locate the very last item in the array. The reason is that when you cut in half, you don't cut perfectly in half. If the array has 16 elements in it, you look first at element 8: there are actually 7 elements above it in the array; but there are 8 elements below it! If the array has any number other than a power of 2, there is always one division which has to be rounded up, and that rounding up gives us room to get to the very end of the array. (Actually, it also caused the problem of going beyond the bounds of the array, which made us add line 55.) There are several ways to overcome the problem, including preventing the array ever from having an "undesirable" number of items. For me, the simplest thing to do is to check the last item in that array if the search fails. If they don't match, then the search actually has failed. But if it does succeed at this point, I do have to assign the value of TL to PT, as PT is what is carried into the main program to tell what item number was found. I do the entire thing in line 70:

70 IF SW$=L1(TL)
THEN PT=TL:GOTO [found it]

I also have to change line 25, so that the GOTO there branches to 70.

If the compare in line 70 yields a false, then the search has really failed, and you drop out of the binary search algorithm. Let's now look at the complete algorithm in figure 3, which is missing only the line numbers after the GOTO statements which will link the search to the programs you use it in.

```
10  PT=INT (TL/2+.5): IV=PT

20  IF L1$(PT)=SW$ THEN GOTO [found it]

25  IF ABS (IV)=1 THEN GOTO 70

30  IV=INT ((ABS(IV))/2+.5)

40  IF L1$(PT)>SW$ THEN IV=-IV

50  PT=PT+IV

55  IF PT  TL OR PT  1 THEN IV=-IV: PT=PT+IV

60  GOTO 20

70  IF  SW$=L1$(TL)  THEN  PT=TL:  GOTO
    [found it]

80  REM Search has failed and you're out
    of the binary search algorithm.
```

Figure 3

THE Sorcerer's Grimoire

the pages turned by Ian Macmillan, member of SCUA, and editor of its newsletter.

The little troll honed his beloved chopper for the thousandth time as he boredly swung his legs over the snake-pit. The Sorcerer was explaining BCD arithmetic. "Binary addition is easy..." He raised his voice over a sudden uproar from the crocodile pool. "...But decimal adders-"

He stopped in amazement. The troll had leapt into the pit and was laying about him with the chopper, counting to ten with every stroke. The Sorcerer was almost speechless. "Stop!" he cried.

The troll realised all was not well. "You did say decimate the adders?" he asked, anxiously. The Sorcerer did not, and would the troll kindly go to the courtyard, set the rustic table, and bring it here on the double! The troll, not loath to leave, left on the run.

In due course the table wobbled into view, the troll revealed only by sundry grunts and gasps from underneath.

"Into the pit with it!", commanded the Sorcerer.

The troll hoisted the table on his shoulder, clambered into the pit, and having placed the table in the centre, leapt with one mighty bound to his favourite perch, where he started to hone his chopper.

I was curious. "Why the table?", I asked.

The Sorcerer seemed surprised at my ignorance.

"Surely you know", he remarked. "The best way to set adders to multiply is to use a log table..."

BINARY CODED DECIMAL ADDERS

The Z-80 processor is equipped with special instructions for decimal arithmetic in BCD format, rarely found in hobby type programs. However they are often used in machine control applications, where inputs from thumbwheel switches, and outputs to displays are usually in BCD form.

The most important 'BCD' instruction is 'DAA', a single byte instruction (27hex) that follows complex rules to adjust the result of the addition, subtraction, increment or decrement of two BCD numbers to be correctly represented in BCD. For example; in BCD, 47 + 53 must yeild 00 plus a carry, but as a hexidecimal sum, the answer is 9A, with no carry.

Note that two decimal BCD digits are held in a single byte. Multiple digits are mostly handled by doing sums on pairs of bytes, with any carry used in the calculation on the next pair, and so on. However some use can be made of the limited 16 bit arithmetic available for the HL and DE registers.

A FOUR DIGIT DECIMAL COUNTER

The following demonstration program implements a counter that having been preset to a value, counts down to zero, resets itself to the initial value, and starts again.

The BCD to ASCII conversion routines demonstrate another 'BCD' instruction, RRD. This rotates the low nybble of the byte in the A register through the byte at the address pointed to by the HL register. Another instruction, RLD does much the same thing, but in the other direction; useful for conversions from ASCII to BCD. Apart from their use in BCD routines, these instructions are very suitable for performing fiendish manipulations in impenetrable programs.

No apologies are made for over elaboration in this program, which has been written to demonstrate some useful techniques and routines with (hopefully) maximum clarity.

Sorcerer monitor routines are used to display the title and clear the screen.

```
;BCD COUNTER DEMONSTRATION PROGRAM
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
;
;MAIN PROGRAM
;------------
0100 CD 11 01      CALL 0111H      ;CALL 'CLEAR SCREEN'
0103 CD 17 01      CALL 0117H      ;CALL 'PRINT TITLE'
0106 CD 47 01      CALL 0147H      ;CALL 'COUNTER & DISPLAY'
0109 CD 2B 01      CALL 012BH      ;CALL 'TIME DELAY'
010C CD 34 01      CALL 0134H      ;CALL 'ESCAPE?'
010F 18 F5         JR   0106H      ;GOTO DECR CNTR; DO IT ALL AGAIN


;SUBROUTINES AND DATA
;--------------------
;CLEAR SCREEN
0111 3E 0C         LD A,0CH        ;GET 'FORM FEED' = CHR$(12)
0113 CD 1B E0      CALL E01B       ;CALL 'VIDEO'; CLEAR SCREEN
0116 C9            RET             ;DONE
;
;PRINT TITLE
0117 21 1E 01      LD HL,011EH     ;ADDR OF START OF TITLE
011A CD BA E1      CALL E1BA       ;CALL 'MESSAGE OUT'
011D C9            RET             ;DONE
;
;ASCII MESSAGE "BCD COUNTER"
011E  42 43 44 20                  ;ASCII "BCD "
```

```
0122  43 4F 55 4E                          ;ASCII   JUN"
0126  54 45 52 3A                          ;ASCII   ER:"
012A  00                                   ;0 IS    ) OF MESSAGE' MARKER
;
;TIME DELAY
0128 11 00 5A      LD DE, 5A00H    ;CONS    FOR 300mS
012E 1B            DEC DE          ;ONE |  .S
012F 7A            LD A,D          ;GET MSB; LSB STILL IN E
0130 B3            OR A,E          ;MSB AND LSB BOTH ZERO?
0131 20 F6         JR NZ, 012EH    ;NO, KEEP LOOPING
0133 C9            RET             ;END OF DELAY
;
;ESCAPE?...IS 'GRAPHIC' KEY PRESSED?
0134 3E 00         LD A,00         ;GET 0 AND SEND TO THE
0136 D3 FE         OUT FE,A        ;KEYBOARD SCAN
0138 DB FE         IN A, FE        ;GET WHATEVER IS THERE
013A E6 1F         AND 01FH        ;MASK OUT ALL JUNK
013C FE 15         CP  015H        ;GRAPHIC KEY?
013E C0            RET NZ          ;NO, SO RETURN
;
013F E1            POP HL          ;YES: GET RID OF RET ADDR
0140 C3 03 E0      JP  E003H       ;AND GO TO MONITOR

;COUNTER PRESET VALUE, AND THE COUNTER
0143 12 34        DATA             ;PRESET VALUE FOR COUNTER: 1234
0145 01 01        COUNTER          ;THE ACTUAL COUNTER WORKS HERE
;
;DECREMENT COUNTER
0147 21 46 01     LD HL, 0103H     ;POINT LOW PR OF COUNTER DIGITS
014A A7           AND A            ;CLEAR CARRY FLAG
014B 7E           LD  A, (HL)      ;GET LOW PAIR OF COUNTER DIGITS
014C 3D           DEC A            ;MINUS ONE
014D 27           DAA              ;DECIMAL ADJUST
014E 77           LD (HL), A       ;PUT NEW LOW DIGITS BACK
014F 21 02 01     LD HL, 0102H     ;POINT HIGH PR OF COUNTER DIGITS
0152 7E           LD  A, (HL)      ;GET HIGH PAIR OF COUNTER DIGITS
0153 DE 00        SBC A, 00        ;SUBTRACT ZERO...AND THE CARRY
0155 27           DAA              ;DECIMAL ADJUST
0156 77           LD (HL), A       ;PUT NEW HIGH DIGITS BACK
;
;IF ZERO THEN RESET TO VALUE
0157 2A 45 01      LD HL,(0145H)   ;GET ALL FOUR DIGITS OF COUNTER
015A 7C            LD A, H         ;TWO DIGITS IN A.TEST RESULT IS
015B B5            OR A, L         ;ONLY ZERO IF ALL BITS ARE ZERO
015C 20 06         JR NZ, 0164H    ;NOT, SO GO TO DISPLAY
015E 2A 43 01      LD HL, (0143H);ZERO: GET VALUE FROM 143 & 144
0161 22 45 01      LD (0145H), HL;RESET COUNTER TO INITIAL VALUE
;
;DISPLAY ON SCREEN
;--------------------
;INITIALISE POINTERS
0164 2A 45 01      LD HL,(0145H)   ;GET THE WHOLE COUNTER AND SAVE
0167 E5            PUSH HL         ;IT FROM RRD OPERATION
0168 21 45 01      LD HL,0145H     ;ADDR 1ST BCD NRS IN COUNTER
016B 11 00 F1      LD DE,F100H     ;SCREEN LOCATION
;
;CONVERT 2 DIGITS IN 1ST BYTE TO ASCII
016E 3E 33         LD A ,33H       ;A=33  (HL)=Nn
0170 ED 67         RRD             ;A=3n  (HL)=3N= MS ASCII DIGIT
0172 47            LD B,A          ;SAVE 2ND ASCII DIGIT
0173 7E            LD A, (HL)      ;GET 1ST ASCII DIGIT
;
```

Software Source

SHOWROOM: 89 OXFORD STREET, BONDI JUNCTION. (02) 389 6388 — POSTAL: P.O. BOX 364, EDGECLIFF. 2027.

IT'S HERE!

THE **LAST WORD** IN PRINTERS

THE ITOH 8510

JUST CHECK THESE FEATURES:

- BOTH TRACTOR & FRICTION FEED
- FULL 9-WIRE HEAVY-DUTY PRINTHEAD
- 9 x 7 DOT MATRIX WITH DESCENDERS
- FULL **PROPORTIONAL** SPACING
- COMPLETE INBUILT GRAPHICS CHARACTER SET INCLUDING GREEK & MATHS SYMBOLS
- DOT-PROGRAMMABLE GRAPHICS
- UNDERLINED, BOLDFACED, EXPANDED CHARACTERS
- 100 cps LOGIC-SEEKING
- LF SIZE SELECTABLE IN 1/144 ths INCH **UP** or **DOWN**
- MARGIN SETTING CAPABILITY
- HORIZONTAL & VERTICAL TAB SETTINGS
- PROGRAMMABLE 40, 48, 68, 80, 96, 236 CHARS/LINE
- EUROPEAN CHARACTERS SWITCHABLE

ALL THE ABOVE ACCESSIBLE THROUGH ANY STANDARD CENTRONICS 7-BIT INTERFACE

AND NOW CHECK THE PRICE: **$999.00** INCLUDING TAX

Spellbinder™

The **ULTIMATE** wordprocessor

FORMS HANDLER	SELECTIVE SORT
MULTI COLUMN PRINT	GLOBAL EDIT
MAIL-MERGE	DECIMAL TABBING
BOILERPLATE	COLUMNAR ADDITION
ALPHA SORT	GLOBAL SEARCH/REPLACE
NUMERIC SORT	INTERACTIVE PROGRAMMING LANGUAGE
HORIZONTAL SCROLL	ON-SCREEN VERIFY

SUPPORTS ANY SYSTEM WHICH USES THE CP/M OR OASIS OPERATING SYSTEM.

CUTS HOURS OFF OPERATOR TRAINING WITH SIMPLICITY OF USE BEING THE KEY TO THE WORLD'S MOST SOPHISTICATED MICRO WORD-PROCESSOR.

DEALER ENQUIRIES INVITED.

SORCERER

EQUIPMENT

Software Source Pty Ltd stocks a COMPLETE RANGE of software and peripherals for the world's Rolls-Royce of Microcomputers — at a Mini price.

PRINTERS	GAMES
PLOTTERS	ACCOUNTING
MODEMS	DATA BASE
EXPANSION	MAILING LIST
DISK SYSTEMS	WORD PROCESSOR
MEMORY	COMMUNICATIONS
VIDEO MONITORS	EDUCATIONAL

NEW

DISKS	CP/M 2.2 vers 3.0 with
CASSETTE RECORDERS	ADDRESSABLE FLASHING CURSOR
CASSETTE TAPES	CLEAR-TO-END-OF-SCREEN/LINE
S-100 BOARDS	INVERSE CHARACTERS
MUSIC SYNTHESISERS	8 PRINTER DRIVER OPTIONS
	TURN-KEY OPERATION

WRITE TODAY FOR FREE CATALOGUE.

```
!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ
[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
αβγδεζηθικλμνξοπρστυφχψωΔΓΣΛΩ♮√°↑↓←→±∓≥≤≈·⊕∞∴%‰0123456789()+-·*/
```

```
;DISPLAY FIRST 2 DIGITS ON THE SCREEN
0174 12            LD (DE),A       ;PUT 1ST DIGIT ON THE SCREEN
0175 13            INC DE          ;POINT TO NEXT SCREEN LOCATION
0176 78            LD A,B          ;GET 2ND DIGIT
0177 12            LD (DE),A       ;PUT 2ND DIGIT ON THE SCREEN
0178 13            INC DE          ;POINT TO NEXT LOCATION
;
;CONVERT 2ND PAIR OF DIGITS
0179 23            INC HL          ;POINT TO SECOND PR OF DIGITS
017A 3E 33         LD A,33H        ;A=33  (HL)=Nn
017C ED 67         RRD             ;A=3n  (HL)=3N= 3RD ASCII
017E 47            LD B,A          ;GET 4TH ASCII DIGIT
017F 7E            LD A, (HL)      ;GET 3RD DIGIT
;
;DISPLAY 2ND PAIR OF DIGITS
0180 12            LD (DE),A       ;PUT 3RD DIGIT ON THE SCREEN
0181 13            INC DE          ;POINT TO NEXT SCREEN LOCATION
0182 78            LD A,B          ;GET 4TH DIGIT
0183 12            LD (DE),A       ;PUT 4TH DIGIT ON THE SCREEN
0184 E1            POP HL          ;GET THE COUNTER BACK
0185 22 45 01      LD (0145H),HL   ;AND PUT IT IN ITS PLACE
0188 C9            RET             ;DISPLAY COMPLETE
;
;
```

Well, I hope that didnt decimate you...because as a parting tit-bit I want to introduce you to a remarkable program that converts a single Hex digit (0-F) to ASCII:

```
HEXIDECIMAL DIGIT IN A TO ASCII IN A
--------------------------------------------
0000 C6 90         ADD A,90H         ;I'LL PUBLISH THE BEST SET OF
0002 27            DAA               ;REMARKS THAT SUCCINTLY EXPLAIN
0003 CE 40         ADC A,40H         ;HOW THIS WORKS.
0005 27            DAA               ;
```

This ingenious arrangement is said to be due to a certain Bill Byerley of Intel.

```
?
COMMAND: MO 3000 3100 100

COMMAND: DU 100 188


ADDR    0  1  2  3   4  5  6  7   8  9  A  B   C  D  E  F

0100:   CD 11 01 CD  17 01 CD 47  01 CD 2B 01  CD 34 01 18
0110:   F5 3E 0C CD  1B E0 C9 21  1E 01 CD BA  E1 C9 42 43
0120:   44 20 43 4F  55 4E 54 45  52 00 00 11  00 30 1B 7A
0130:   B3 20 FB C9  3E 00 D3 FE  DB FE E6 1F  FE 15 C0 E1
0140:   C3 03 E0 12  34 07 16 21  46 01 B7 7E  3D 27 77 21
0150:   45 01 7E DE  00 27 77 2A  45 01 7C B5  20 06 2A 43
0160:   01 22 45 01  2A 45 01 E5  21 45 01 11  00 F1 3E 33
0170:   ED 67 47 7E  12 13 78 12  13 23 3E 33  ED 67 47 7E
0180:   12 13 78 12  E1 22 45 01  C9
```

MARCH TO A DIFFERENT DRUMMER

The programmable 'rhythm box' has exploded onto the music scene in recent years. Jeff Aughton shows how to make a PET add-on, which outperforms many commercial units, for $20.

This add-on allows you to compose rhythms, store them using the computer and then play them back through a 'drumbox'; that is, a set of circuits capable of imitating conventional percussion instruments. If required, the contents of the computer memory may be stored on tape and reloaded at a later date — this facility is particularly useful for stage work. Software for the project consists of a program, written in Basic for an 8k PET, which provides a series of control pulses at the user port. Hardware requirements are the drumbox itself, whose construction is described later, plus, of course, an amplifier to replay the music. At current prices, the total cost of the project, including connectors, is about $20.00.

Keyboard instruments, such as electronic organs, often include a rhythm section capable of playing a number of preset rhythms, eg, waltz, Latin and disco. Figure 1 shows the layout of a typical unit: rhythm patterns are taken from the memory by the controller which then provides trigger pulses for the instrument generators. When the pattern is complete, the controller resets and starts the cycle all over again. Although this system is cheap and reliable, it suffers from the major drawback that the patterns produced are repetitive and cannot be easily changed. Normally, the controller/memory combination comprises a number of discrete components; in our case it is replaced by the computer.

Early rhythm units used a diode matrix as the memory and, while this method is rather cumbersome, it does at least allow 'reprogramming' with a soldering iron. More recently, custom-designed ICs (effectively ROM chips) have been used to control up to eight instruments for, typically, 16 rhythm patterns.

This project allows control over four instruments — bass drum, snare drum, hi-hat and cymbal — which form the basis of most modern rhythm backing.

We now look in detail at the facilities offered by the program.

Nine rhythms (numbered 1-9!) may be stored at any time. Of these, 1-6 contain 16 beat measures and are suitable for 4/4 and 2/4 time signatures, while 7-8 contain 12 beat measures (for 3/4 and 6/8). Rhythm 9 allows each bar of the music to be individually selected from any bar stored in the first eight patterns. This is obviously the most versatile arrangement and is best suited to recording or experimental work.

Each pattern consists of two bars (A and B) which can be replayed in one of the following modes: A, in which the A bar is repeated continually; AB when the A bar is followed by the B bar and the AB pattern is repeated continually; and fill-in when the A bar is played several times followed by a one bar fill-in of the B bar — this pattern is then repeated.

Part of the screen display is shown in Figure 2. The (moving) arrow indicates the progress of the music and the ● s prepresent beats. Rests are indicated by —s.

The music may be stopped and restarted from the beginning of the first bar by pressing the space key. Pressing any key other than space will cause the program to reset to the selection menu — note that this will not affect the memory in any way, as the rhythms can only be changed when the program is in the 'write' mode.

Storing the memory contents on tape, or loading patterns from tape, is done simply by selecting the correct mode ('file' and 'load' respectively) and following the instructions contained in the program.

Before looking at how the program works, we will consider briefly the operation of the user port. This is the central set of connectors at the rear of the machine (Figure 3) and the input/output port itself comprises pins C through L.

The address of this port is 59471 decimal and POKEing this with, for example, 13 (=00001101 binary) would cause pins C,E and F to go high (+5 V) assuming that these lines are configured as outputs. In this project, only pins C to F are used (see Table 1) and these four channels are set as output lines by POKEing 59459 with 15 (=00001111).

Thus, POKEing 59471 with 13 (= 8+4+1) would cause the cymbal, hi-hat and bass drum to sound simultaneously.

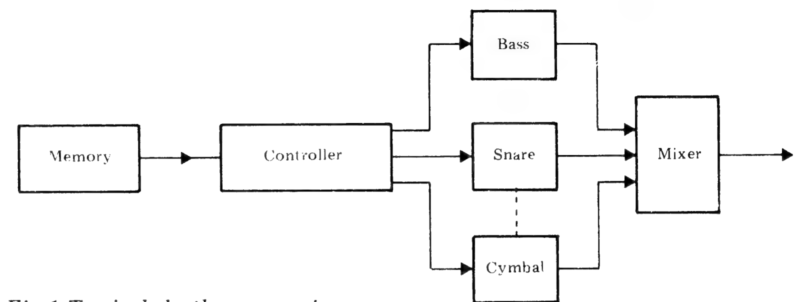


Fig 1 Typical rhythm generator

*** Rhythm 1 ***

Pattern A

Bass	●	–	–	–	–	–	–	●	●	–	●	–	–	–	–	–
Snare	–	–	–	–	●	–	–	–	–	–	–	–	●	–	–	–
Hi-hat	●	–	●	–	●	–	●	–	●	–	●	–	●	–	●	–
Cymbal	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Pattern B

Bass	●	–	–	–	–	–	–	●	–	–	●	–	–	●	–	–
Snare	–	–	–	–	●	–	–	●	●	●	–	●	●	–	–	–
Hi-hat	●	–	●	–	●	–	●	–	–	–	–	–	–	–	–	–
Cymbal	–	–	–	–	–	–	–	–	–	–	–	–	–	●	–	–

Fig 2 Screen layout

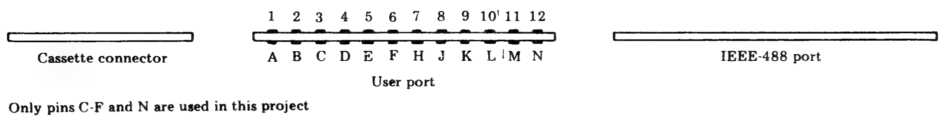


Fig 3 Rear view of PET ports

Instrument	Triggered by POKEing port with	Pin
Bass drum	1	C
Snare drum	2	D
Hi-hat	4	E
Cymbal	8	F

Table 1 I/O port use

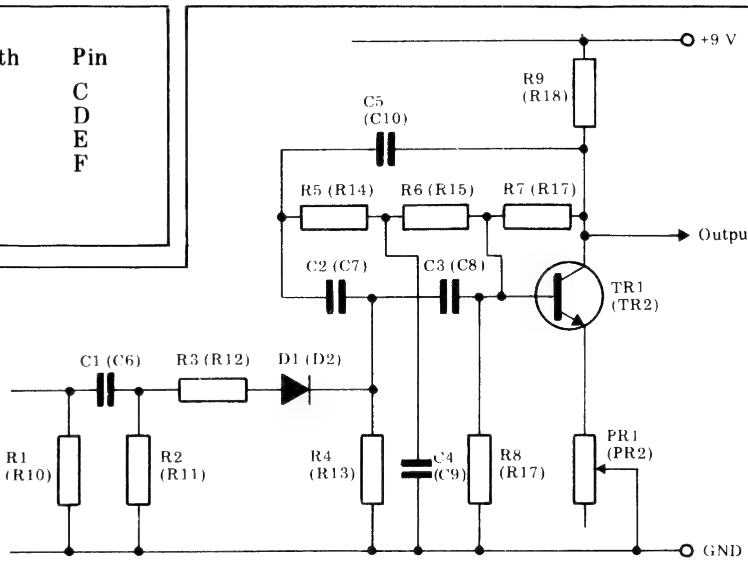


Fig 4 Bass drum. The snare drum uses the same circuit with the component number in brackets.

SNARE DRUM 2

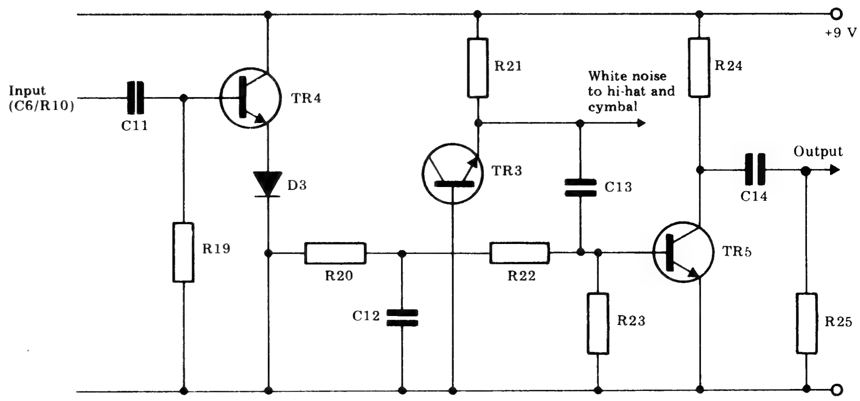


Fig 5

The actual POKEing is done at line 4020. Notice that the lines are immediately POKEd off at line 4030. This is because the instrument generators only need very short pulses to trigger them.

Within the program, one of the most important features is data storage. Data is held in two arrays:

1. B%(32,8) These are integers in the range 0-15 which are POKEd into the user port, thus triggering the various sounds. The '8' denotes the eight rhythms and the '32' represents the contents of the two bars associated with each rhythm (1-16 for A and 17-32 for B);
2. C%(200) This array contains the contents of 'rhythm 9' that is, the sequence of bars to be played for that rhythm as chosen by the operator.

One unusual feature of the program is the way in which the contents of the screen are written to memory. This occurs at line 5190 onwards and the computer actually reads the line as seen on the screen (in the same way that the Basic interpreter does each time a program line is entered). This means that however many changes are made during editing, the line is read *once* and stored only when RETURN is pressed.

As stated, the purpose of this program is to provide a series of short positive-going trigger pulses at pins C—F. To test the program properly obviously requires the drumbox itself, but a preliminary test is possible at this stage.

Load the program, type in the rhythm pattern shown in Figure 2 and then replay it (see instructions later in the article). A'l should be visually correct

To check the operation of the I/O port, add line:

4025 FOR J = 1 TO 1000 : NEXT

and re-run the program. Now, during playback, a 0-10 V meter connected between pins N (—ve lead) and one of C—F (+ve lead) should register approximately 5 volts in time with the moving arrow. Thus, when a bass drum beat is present, pin C should go high, etc. This test should be carried out carefully as the PET is rather delicate ir this area and is not tolerant of electrical errors on the user port.

Now, if all is well, delete line 4025, switch off the computer and plug in your soldering iron — to complete the project the drumbox has to be constructed.

Hardware

The instrument generators and mixers are mounted in Veroboard and housed in a small plastic box. Although the layout is not critical, the unit should not be placed too near to transformers (or computers!) or other possible sources of interference. If this is unavoidable, the circuit-board should be mounted in a screened metal box. The unit is powered by a 9 V battery.

The generators themselves are of two types: pitched sounds such as those of a bongo or bass drum are produced by the damped oscillations of a 'twin-T' oscillator, while unpitched sounds such as the cymbal use shaped and filtered white noise.

The snare drum uses one circuit of each type to produce the two distinct tones which make up the snare sound — oscillations for the 'plonk' and white noise for the 'tizz'.

Both of the 'twin-T' oscillators behave in the same way, the only real difference between them being the frequency-determining capacitors in the Twin-T networks, so that the snare drum is pitched approximately two octaves above the bass drum.

In Figure 4 PF1 adjusts the gain of the circuit and is set so that the oscillator is held just short of resonance. When an input pulse is received, it is differentiated by C1/R2 and the negative-going edge (when the pulse is removed) is eliminated by D1. The remaining short pulse causes the circuit to oscillate, but since the oscillations are damped, the waveform produced decays rapidly, thus producing the required envelope for the sound (see Figure 9).

White noise is produced by reverse-biasing the base-emitter junction of a transistor, TR3. Noise is taken from the emitter and fed to the three noise-shaping circuits. Note that the amount of noise produced depends on the transistor and it may be necessary to experiment with different transistors, or to adjust the value of R21, to achieve the right effect. To avoid too much soldering, a transistor socket could be mounted in place of TR3 to make it easy to change transistors. The first transistor I tried was a BC108A, but most small-signal NPN transistors stand a chance of working.

Taking the cymbal as an example Figure 8, the input pulse charges capacitor C18 via diode D4 and the decaying voltage at C18/R29 junction controls the level of the white noise through the transistor.

DIFFERENT DRUMMER

The decay time depends on the value of the capacitor and is very short for the hi-hat and longer for the cymbal. The shaped noise appearing at the collector of TR7 is filtered before passing to the mixer.

The mixer is based around TR8. The values of the input resistors R33-R37 determine the relative balance of the instruments and some experimentation may be needed to achieve the best effect. Increasing the value of a resistor decreases the level of that instrument. Alternatively, these resistors could be replaced by 470k presets, thus allowing the instrument balance to be varied at will. As space was limited in the prototype, this modification was not included.

Construction

A Veroboard layout is shown in Figure 10. Assemble the components, with the exception of R35, noting carefully the orientation of diodes and transistors and that the copper track is broken in the correct places. After the board has been constructed, check that all components are in the right place and that there are no solder bridges between adjacent tracks of the board.

Rotate PR1 and PR2 fully clockwise and connect the board to a fresh 9 V battery and to an amplifier via a screened lead. At this stage, nothing should be heard from the amplifier. Slowly rotate PR1 anticlockwise. At a certain point a continuous low frequency sine wave will be heard. PR1 should be set just below the point where oscillation occurs.

Temporarily attach a lead to the battery positive and briefly touch the other end to the bass drum input (R1/C1). The sound of the bass drum should now be heard. It is possible to make small adjustments to PR1 to damp the sound of the drum in the same way that a real drummer will pad the inside of the drum to produce the sound he most prefers.

Repeat the process for the snare drum, adjusting PR2. Notice that the sound is higher pitched than the bass drum and that this is only part of the snare sound (ie the sound that a drummer gets by disconnecting the snare).

Resistor R35 can now be fitted to the board. On retriggering, the snare drum should sound much more realistic now that the sound of white noise has been added.

Triggering the cymbal and hi-hat should produce rushes of white noise, with a noticeable decay time on the cymbal envelope. Notice that, in practice, the unit is triggered by a very short pulse from the computer and that these sounds begin to decay almost as soon as they are switched on.

If these tests are satisfactory, switch off and wire the unit to the user port connector as shown in Figure 11a. A 100 uF 16 V electrolytic capacitor is shown connected across the switch terminals; the siting of this capacitor is not critical — there is simply no room for it on the circuit-board.

Final adjustments, if required, can be made after the generator has been connected to the computer.

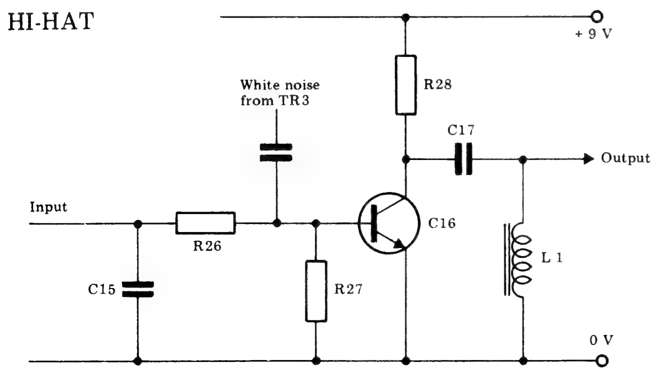


Fig 6

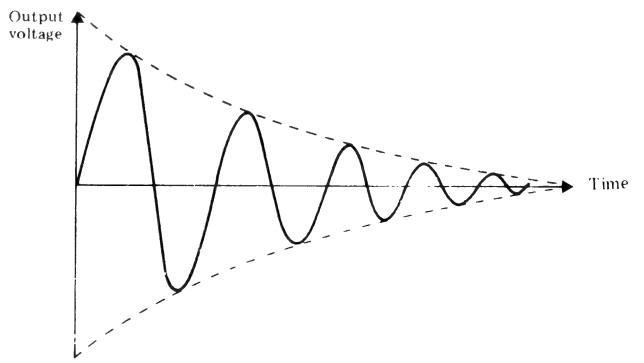


Fig 9 Damped oscillations produced by 'Twin-T' circuit

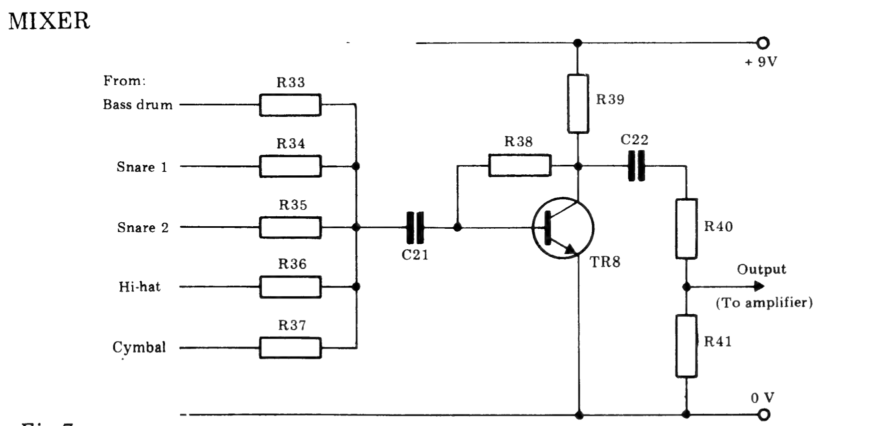


Fig 7

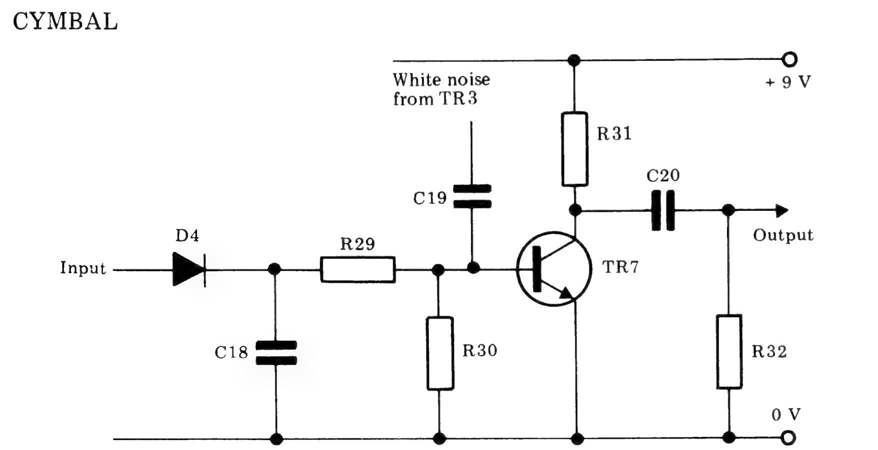


Fig 8

Using the unit

Connect the drumbox to a suitable amplifier (see note later) and to the user port. Load and run the program and select the 'write' option (1) as, at this stage, no rhythms have been stored. Select Rhythm 1, a 4/4 pattern. The contents of Rhythm 1 (blanks) will be written to the screen. Write Rhythm 1 using the pattern shown in Figure 2, and the following keys: Space to write a rest (-) in the music; Delete to Delete the last beat before the cursor; and Return to move the cursor to the next line or recall the selection menu after the last line has been written. Any other key writes a beat (.) in the music. Note that after return is pressed, no more editing is possible on the line just written as the cursor passes to the next line. Now select the 'read' option (2), select Rhythm 1, select 'fill-in'

mode (3) and select a fourth bar fill-in (1). Choose a tempo in the range 80 to 85 and press space.

The moving arrow shows the progress of the music which will now be playing. After three bars of A, the arrow will move to bar B and play that pattern for one bar. The whole process now repeats.

The rhythm can be stopped and restarted from the beginning of bar A by pressing space. To stop the run and recall the selection menu, press any key (not 'stop!) other than space while the rhythm is playing. This rhythm pattern will remain in the computer until it is switched off or until it is overwritten with a new Rhythm 1.

To edit the pattern, call it with 'write' and make the necessary changes. For example, to add a cymbal to the first beat of bar B, press return seven

```
100 REM DIGITAL DRUMMER
110 REM
120 REM BY J.AUGHTON
130 REM
140 DIMB%(32,8),C%(200)
150 L1=33455:L2=33735:P=59471
160 POKE59459,15
170 PRINT"▯"
180 REM
200 REM **** MENU ****
210 POKEP,0:GOSUB2000
220 PRINT"   1) WRITE"
230 PRINT"   2) READ"
240 PRINT"   3) FILE"
250 PRINT"   4) LOAD"
260 PRINT"   5) QUIT"
270 PRINT"▯WHICH MODE? ";
280 X=5:GOSUB3000
290 ONXGOTO5000,6000,7000,8000
300 POKE59459,0
310 PRINT"▯":END
320 REM
1000 REM **** DRAW BOARD ****
1010 PRINT"▯▯▯▯▯▯▯▯▯▯▯           *** RHYTHM";R;"***▯"
1020 V=0:FORI=65TO66:RESTORE:X=1
1030 PRINT"▯▯ PATTERN ";CHR$(I);"
1040 PRINT:READA$:PRINTA$;
1050 FORJ=1TO16
1060 IFB%(J+V,R)ANDXTHENPRINT"● ";:GOTO1080
1070 PRINT"- ";
1080 NEXTJ
1090 X=X*2:IFX<9THEN1040
1100 V=16:PRINT:NEXTI
1110 RETURN
1120 REM
1500 REM **** WAIT... ****
1510 GETA$:IFA$<>" "THEN1510
1520 RETURN
1530 REM
2000 REM **** CLEAR TOP ****
2010 PRINT"▯";:FORI=1TO9
2020 PRINT"
2030 NEXT:PRINT"▯";:RETURN
2040 REM
3000 REM **** INPUT ****
3010 POKE158,0:POKE167,0
3020 GETA$:IFA$=""THEN3020
3030 IFA$<"1"ORA$>"9"THEN3020
3040 IFVAL(A$)>XTHEN3020
3050 X=VAL(A$):PRINTX
3060 POKE167,1:RETURN
3070 REM
3500 REM **** TEMPO ****
3510 GOSUB2000:PRINT"▯▯TEMPO 1(SLOW)-99(FAST)";
3520 INPUT"▯▯.▯▯▯▯";A$
3530 S=VAL(A$):IFS<1ORS>99THEN3510
3540 S=397-4*S
3550 PRINT"▯USE ▯ SPACE ▯ TO START AND STOP"
3560 PRINT"▯USE ▯ R ▯ (DURING RUN) TO RESET"
3570 RETURN
3580 REM
4000 REM **** PLAY ****
4010 FORI=1TOV:POKEL,30
4020 POKEP,B%(I+X,R)
4030 POKEP,0
4040 GETA$:IFA$=""THEN4070
4050 IFA$=" "THENPOKEL,32:N=2:RETURN
4060 N=3:RETURN
4070 FORJ=0TOS:NEXTJ
4080 POKEL,32:L=L+2:NEXTI
4090 RETURN
4100 REM
5000 REM **** WRITE ****
5010 L=33293:S=1:X=9:Y=0
5020 PRINT"▯WHICH RHYTHM (1-9)? ";
5030 GOSUB3000:R=X
5035 IFR=9THEN5500
5040 GOSUB1000:V=16:IFR>6THENV=12
5050 PRINT"▯▯▯▯▯▯▯▯▯▯▯▯"
5060 T=1:PRINT:PRINT"▯▯▯▯▯▯▯▯";
5070 POKE167,0:POKE158,0
5080 GETA$:IFA$=""THEN5080
5090 X=ASC(A$)
5100 IFX=20THEN5170
5110 IFX=13THEN5190
5120 IFX=32THEN5150
5130 IFT<V+1THENPRINT"● ";:T=T+1
5140 GOTO5070
5150 IFT<V+1THENPRINT"- ";:T=T+1
5160 GOTO5070
5170 IFT>1THENPRINT" ←←← -←";:T=T-1
5180 GOTO5070
5190 REM ENTER
5200 IFPEEK(PEEK(196)+256*PEEK(197)+PEEK(198))>128THEN5200
5210 POKE167,1
5220 FORI=1TO16
5225 B%(I+Y,R)=B%(I+Y,R)ANDNOTS
5230 IFPEEK(L+2*I)=45THEN5240
5235 B%(I+Y,R)=B%(I+Y,R)+S
5240 NEXT
5250 L=L+40:S=S*2:IFS<9THEN5060
5260 L=L+120:S=1:Y=16
5270 IFL>33727THEN200
5280 PRINT"▯▯":GOTO5060
5290 REM
5500 REM **** WRITE RHYTHM 9 ****
5510 PRINT"▯SELECT EACH BAR OF THE MUSIC FROM THOSE▯"
5520 PRINT"ALREADY PROGRAMMED.INPUT EACH BAR IN THE"
5530 PRINT"FORM 1A- MEANING RHYTHM 1/PATTERN A ETC."
5540 PRINT"FINISH THE LIST BY TYPING XX▯▯"
5570 B=0
5580 IFB<200THEN5610
5590 PRINT"▯YOU HAVE WRITTEN 200 BARS-THAT IS THE▯"
5600 PRINT"MAXIMUM ALLOWED":GOTO200
5610 B=B+1:PRINT"BAR #";B;
5620 INPUT"▯▯.▯▯▯▯";A$
5625 IFLEN(A$)<>2THEN5680
5630 IFA$="XX"THENC%(0)=B-1:GOTO170
5640 X=ASC(A$)-48
5650 IFX<1ORX>8THEN5680
5660 B$=RIGHT$(A$,1)
5670 IFB$="A"ORB$="B"THEN5690
5680 PRINT"▯ ** ERROR-TRY AGAIN **▯":GOTO5610
5690 IFB$="B"THENX=-X
5700 C%(B)=X:GOTO5580
5710 REM
6000 REM **** READ ****
6010 PRINT"▯WHICH RHYTHM (1-9)? ";
6020 X=9:GOSUB3000:R=X
6030 IFR=9THEN6500
6040 GOSUB1000:V=16:IFR>6THENV=12
6050 GOSUB2000:PRINT" 1) A"
6060 PRINT" 2) AB"
6070 PRINT" 3) FILL-IN"
6080 PRINT"▯WHICH MODE? ";
6090 X=3:GOSUB3000
6100 ONXGOTO6110,6300,6400
6110 Q=1E9
6120 GOSUB3500
6190 GOSUB1500
6200 FORK=1TOQ
6210 N=1:L=L1:X=0:GOSUB4010
6220 ONNGOTO6230,6190,200
6230 NEXTK
6240 N=1:L=L2:X=16:GOSUB4010
6250 ONNGOTO6200,6190,200
6300 Q=1:GOTO6120
6400 REM FILL-IN
6410 GOSUB2000:PRINT"FILL IN AT:▯
6420 PRINT" 1) 4TH BAR
6430 PRINT" 2) 8TH BAR
6440 PRINT" 3) 16TH BAR
6450 PRINT"▯ WHICH? ";
6460 X=3:GOSUB3000
6470 Q=2↑(X+1)-1:GOTO6120
6480 REM
6500 REM **** READ RHYTHM 9 ****
6510 GOSUB3500:GOSUB1500
6515 PRINT"▯             RHYTHM 9"
6520 FORK=1TOC%(0)
6525 PRINT"▯   ▯";K
6530 N=1:X=0:IFC%(K)<0THENX=16
6540 L=4E4:R=ABS(C%(K)):V=16+4*(R>6)
6550 GOSUB4010:ONNGOTO6560,6600,200
6560 NEXTK:GOTO200
6600 GOSUB1500
6610 GOTO6520
6620 REM
7000 REM **** FILE ****
7010 PRINT"▯PLACE A BLANK TAPE IN THE CASSETTE UNIT"
7020 PRINT"▯HIT SPACE WHEN YOU ARE READY"
7030 GOSUB1500
7040 OPEN1,1,1
7050 FORI=1TO32:FORJ=1TO8
7060 PRINT#1,B%(I,J)
7070 NEXTJ,I
7075 FORI=0TOC%(0):PRINT#1,C%(I):NEXT
7080 CLOSE1:PRINT"▯FILING COMPLETE"
7090 FORI=1TO2000:NEXT
7100 GOTO170
8000 REM **** LOAD ****
8010 PRINT"▯PLACE A DATA TAPE IN THE CASSETTE UNIT"
8020 PRINT"▯HIT SPACE WHEN YOU ARE READY"
8030 GOSUB1500
8040 OPEN1
8050 FORI=1TO32:FORJ=1TO8
8060 INPUT#1,B%(I,J)
8070 NEXTJ,I:INPUT#1,C%(0)
8075 FORI=1TOC%(0):INPUT#1,C%(I):NEXT
8080 CLOSE1:PRINT"▯LOADING COMPLETE"
8090 GOTO7030
9000 DATA "BASS    ","SNARE   ","HI-HAT  ","CYMBAL  "
```

DIFFERENT DRUMMER

times (the first seven lines are correct and need not be rewritten) and then any key (to write the beat) followed by return.

When the rhythms have been written they may be saved on a data tape. To do this, select the 'file' mode (3) and follow the instructions provided. Rhythms may be retrieved from the tape by selecting the 'load' mode (4). The contents of Rhythm 9 (if any) will also be saved by the file command. With a little practice, the unit becomes very easy to use.

Readers who are into, say, 7/4 rhythms (although personally, I have enough trouble with 4/4) can change: IF R)6 THEN V=12 in lines 5040 and 6040 to:
IF R) 6 THEN V=14
This will produce 14 beat measures in Rhythms 7 and 8 and should enable them to indulge in their own particular whims.

To ensure successful operation of this project, the following points should be noted: The amplifier/speaker combination should be capable of handling the low frequencies generated by the bass drum. An amplifier of at least 20 watts feeding a 12in (or more) speaker is recommended. Further, the character of the sound produced depends very much on the settings of the amplifier tone controls and some experimentation will be needed to produce the best sound. If an 'earth-loop' occurs — which is apparent by the noise from the amplifier when the unit is not playing (or even when it is switched off) — disconnect the amplifier mains lead in the mains plug. *Don't*

Components list

Resistors (all 5%, ¼W)		Capacitors (disc or polyester)	
R1, R4, R10, R13, R39	10k	C1, C4	150nF
R2, R3, R11, R12,	100k	C2, C3, C6, C12, C19, C13	47nF
R5, R6, R14, R15.	68k	C5, C9, C10	33nF
R7, R16, R22, R23, R26, R27, R29, R30, R36	1M	C7, C8, C15	10nF
R8, R17, R37	470k	C11, C21, C22	100nF
R9, R18	56k	C14, C16	2nF
R19	150k	C17, C20	1nF
R20	3k3	C18	330nF
R21	47k	C23 (see text)	100uF, 16 V W electrolytic
R24, R28, R31, R40	4k7	Semiconductors	
R25	22k		
R32	15k	D1-4	1N4148
R33	1M8	TR1, 2, 4, 5, 6, 7, 8	BC108C
R34, R38	2M2	TR3	BC108A
R35	270k		
R41	1k	Misc	
PR1, PR2 horizontal-mounting presets	2k5	L1	100 mH
		Battery, connectors, on-off switch, wire, Veroboard, screened lead	

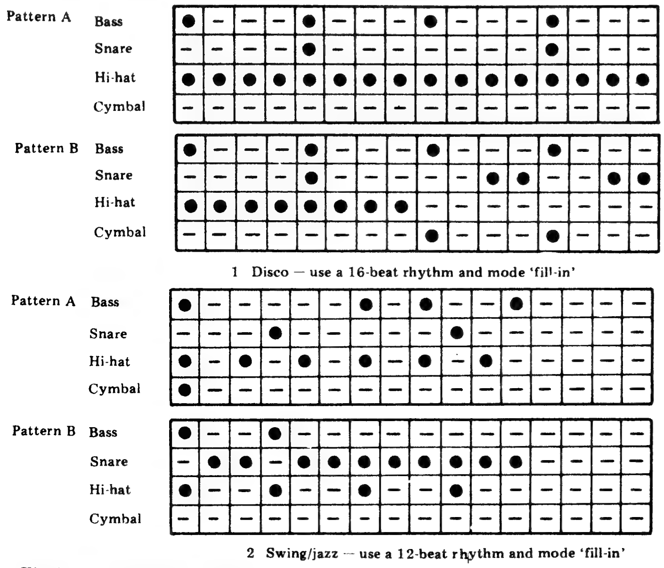


Fig 12 Two sample rhythms

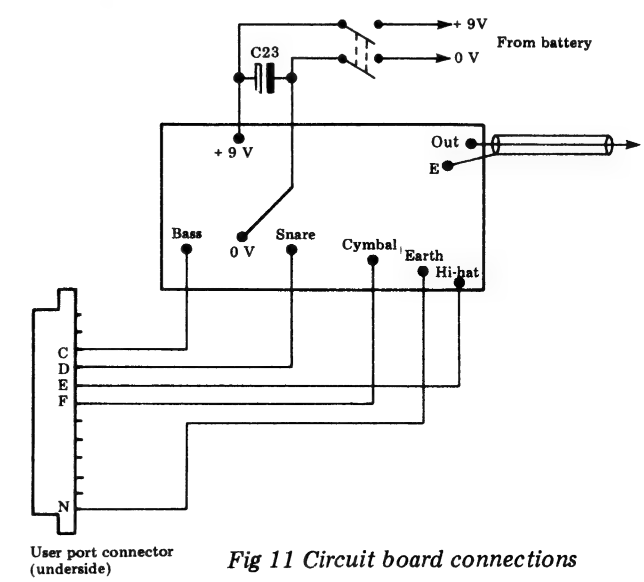
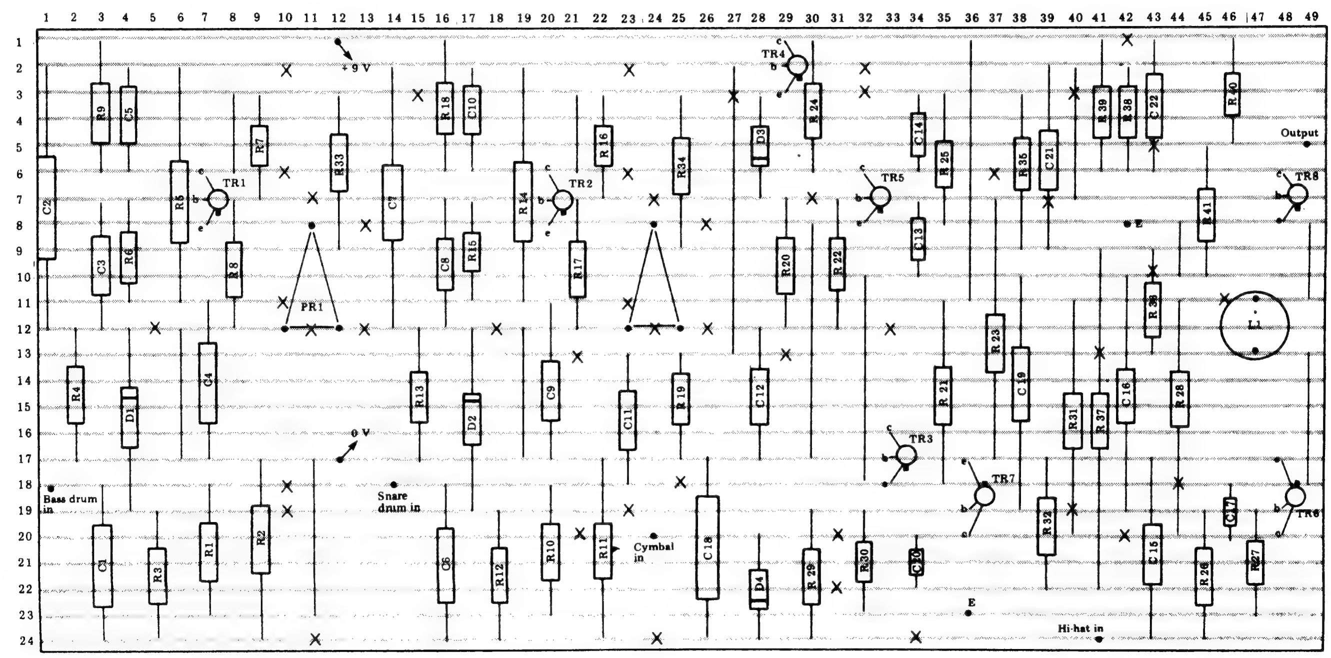


Fig 11 Circuit board connections



NB. it is advisable to install the 13 wire links before fitting any of the components
→ Direction of copper strips X represents a break in the copper track

Fig 10 Vero board layout of the rhythm generator

FREE TRIP TO MELBOURNE

Computer Country is now proud to announce its "FREE TRIP TO MELBOURNE" plan. This plan applies only to interstate purchasers (non-Victorian) of a NEC PC-8000 Computer System or an Hitachi Peach Computer Package.

(Note — Plan applies only to those systems having a minimum configuration value of $4,000 retail without sales tax.)

This is the perfect opportunity to purchase either one of some of the best computers systems on the microcomputer market. You can just buy hardware or add on any one of a number of software packages available which include word processing, general accounting, statistics, entertainment and educational.

THE NEC PC—8000 PROFESSIONAL COMPUTER

It combines the most wanted features of existing micros with new features you have always been looking for.

These features include 4MHx Z-80 CPU, 80 x 25 display with graphics and 8 colours, 32K ROM, parallel/serial/cassette interface, upper/lower case, numeric keypad, 10 special function keys and real time clock — all built-in.

The NEC Professional Computer gives you much more for your money than almost any other micro on the market.

The NEC PC-8000 is supported nationwide by the Hanimex group of companies.

HITACHI "PEACH"

One of the most powerful new microcomputers on the market which uses the 6809 microprocessor. Comes with excellent super high resolution graphics — 640 x 200 dots and has a 40/80 character x 25 lines user programmable screen configuration.

Either choice will give you a great price, a great computer system, a free trip to Melbourne and a special training session as well.

All in all — a great deal from Computer Country.

COMPUTER COUNTRY PTY. LTD.

338 Queens St, Melbourne, Victoria 3000, (03) 329 7533.

Note — Those residing within the Sydney—Adelaide radius will receive 2 economy air tickets from those capital cities. Outside that radius, e.g. Perth, Darwin and Brisbane, this offer is good for one economy air ticket from those cities.

INSIDE THE INTERPRETER

Understanding how your Basic interpreter works can speed up your programs. A F T Winfield shows how.

The command 'RUN' is a sort of incantation among computer users — it is the word that magically transforms a couple of hundred Basic statements into that new space invaders game, or flight simulator! But what happens when you type RUN? Why are some Basic interpreters slower than others? How can I speed up my programs? In this article, I shall take a peek inside a number of Basic interpreters, to try and throw some light upon these questions.

Interpret?

Firstly, what *is* a Basic interpreter? Well, most interpreters contain two parts: an editor and an interpreter (although the two are lumped together under the term 'interpreter'). The editor allows you to enter the program into the machine and modify it if necessary, but the interpreter takes over completely when you type RUN. The interpreter then reads the program from memory, statement by statement, examining each statement to identify reserved words, check syntax, and determine the operation to be performed by the statement, and then it actually carries out the operation.

A feature of the Basic language is that all statements start with a reserved word (the only exception is in the case of a LET statement where the LET may often be omitted). This feature simplifies RUN time processing considerably, since the interpreter need only identify the first word in the statement — to determine the operation to be performed by the statement. Thus, a flowchart for the RUN-time part of an interpreter might be as in Figure 1.

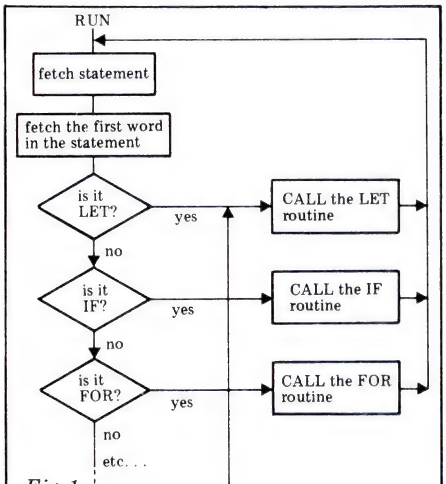


Fig 1

The word LET is tested for first, because it is the most frequently occurring statement in Basic programs, IF is the second most frequently occurring statement and so on. PRINT and other I/O statements are usually low down in the list because speed of execution is limited by the input/output peripherals, not the interpreter.

If the interpreter cannot identify the first word in the statement, then it assumes the statement is a LET statement, without the LET. Thus in interpreters of this kind the statement LET A=42, say, is executed much more quickly than the statement A=42.

This outer flowchart for the interpreter contains no syntax-checking — all of the syntax checking is done in the routines for the individual statements. This can cause some peculiar error messages if you have mistyped the first word in the statement. For example,

PRIMT A

is likely to cause the error message 'missing ='. (Although most interpreters will produce the distinctly unhelpful message 'syntax error'!)

The LET statement

The flowchart for the LET routine might be as in Figure 2.

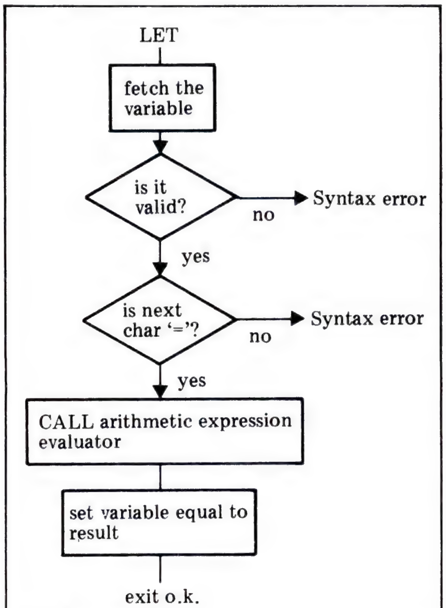


Fig 2

Other statements than LET may, of course, include arithmetic expressions — and so the arithmetic expression evaluator is likely to be a sub-routine which is called by a number of different statement routines. By far the largest part of the time spent during execution of most programs is in evaluating arithmetic expressions — and so understanding how an expression evaluator works can often result in worthwhile speed improvements, as I hope to show later in this article.

But now an example from Denver Tiny Basic [1]; Denver is the smallest Basic interpreter I have come across, being a little over 2 kbytes! It has a structure similar to the one I have just described and no pre-processing by the editor, so that programs are stored in memory exactly as they are typed in.

The program:

```
10 A=1
20 A=A+1
30 IF A<1000 GOTO 20
40 END
```

executes in 8.9 seconds on a 4 MHz Z80. But replace line 20 by LET A=A+1 and the execution time reduces to 7.4 seconds. An improvement of over 10 percent!

Tokens and links

Most extended Basic interpreters *do* pre-process the program as it is entered in two important ways: tokenising and link listing.

Tokenising means replacing each reserved word in the program by a single byte token, for example, Crystal Basic 2.2, would store the line,

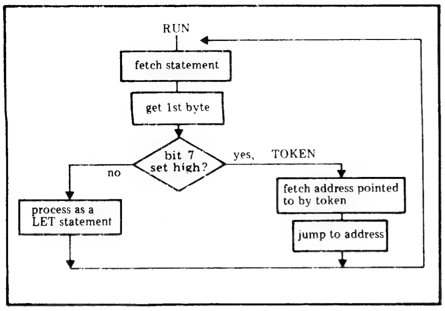
10 LET A=1

(starting at memory address 2D00) as,

```
2D00: 09  2D  link
      OA  00  10
      88      LET
      41      A
      B0      =
      31      1
      00      terminator
```

with the reserved words LET and '=' replaced by the tokens 88, and B0 respectively. So that tokens are never confused with non-tokenised parts of the program, like 'A' and '1', all tokens have bit 7 set high, and all ASCII characters have bit 7 set low.

Apart from the minor benefit of conserving memory, the objective of tokenising is to simplify the RUN time processing, and therefore speed up execution. This is achieved by using the tokens, at RUN time, to point into a table of addresses of statement routines. Let me illustrate this with a new RUN flowchart:.



The ability to process LET statements without the LET still exists here, but often in interpreters of this type, LET statements without the LET are now marginally faster!

The RUN time speed improvement is, of course, achieved at the expense of editor complexity, since the editor must now perform all of the complex ASCII string comparisons, to identify reserved words. Also the LIST command must de-tokenise the program to make it readable!

An interesting aside is that tokenising is *not* the same as syntax checking. In fact, surprisingly few Basic interpreters perform true syntax checking on program entry. Surprising because this feature is often hailed as one of the major advantages of 'interactive' languages! One of the few popular languages that does is the ubiquitous ZX81 Basic interpreter, which stubbornly refuses to accept an ill-formed line! The syntax checking is simplified considerably in this machine by the single word 'keyword' entry — which means that tokenising is done automatically. Some of the hairiest problems of syntax checking like, for example:

```
10 LET BTO=1
20 FOR ATO=BTO TO 100 (!!)
```

are alleviated because the keyword TO is unambiguously defined by the single key TO entry. Despite the syntax checking, the ZX81 internal representation of the program is identical to the [link address], [line number], [tokenised line] format already described. Arithmetic expressions are stored unaltered (apart from tokenisation), and are then still subject to the RUN-time speed improvements I describe in this article.

Link listing is the second technique for speeding up program execution. A linked list Basic program is one in which the start of each line contains a pointer to the start of the next line. Looking again at the internal representation of the example in Crystal Basic, the very first pair of bytes in the stored line make the address 2D09, which point to the start of the next line in the program. The pair of bytes immediately after the link address are the line number, in binary, and to find any given line in a program means simply skipping directly from link to link, comparing the following line numbers, until the right one is found. GOTO statements are then very much faster in interpreters of this type.

Denver Tiny Basic executes the program:

```
10 A=1
20 A=A+1
30 GOTO 40
40 IF A<1000 GOTO 20
```

in 10.7 seconds. Remove the superfluous GOTO in line 30, and the execution time becomes 8.9 seconds. Crystal Basic executes these two programs in 6.1 and 5.6 seconds respectively, a smaller proportional improvement because of the linked list storage of the program in Crystal and, hence faster GOTOs. It is worth remembering that superfluous GOTOs take up significant execution time — and should be avoided.

The GOSUB statement executes in a similar way to the GOTO, with a search through the linked list for the destination line number. The time taken to execute a GOSUB then depends, like the GOTO, largely upon the position of the destination line in the program — particularly if the program is a large one. An example will illustrate this — the Crystal program:

```
10 FOR A=1 TO 1000
20 GOSUB 1000
30 NEXT A
40 END
50 REM
51 REM
   . . . 50 lines altogether
1000 RETURN
```

executes in 3.9 seconds. But place the subroutine at the start of the program:

```
10 RETURN
20 REM
21 REM
   . . . 50 lines
1000 FOR A=1 TO 1000
1010 GOSUB 10
1020 NEXT A
```

and execute by typing 'GOTO 1000', and we see an execution time of 1.9 seconds — a dramatic halving of execution time just by rearranging the program! In fact, amazing speed improvements can often be made simply by placing the subroutines at the start, and the main body of the program at the end, rather than the usual practice of subroutines at the end!

Arithmetic evaluation

As I mentioned earlier, the average Basic program spends most of its time evaluating arithmetic expressions — particularly if floating point, or, worse still, trigonometric functions are involved. Some examples using Crystal Basic will illustrate this:

The program:

```
10 FOR A=1 TO 1000
20 LET B=1
30 NEXT A
```

executes in 2.0 seconds. Replacing line 20 by LET B=1*1 slows the program down to 3.1 seconds and replacing line 20 again by LET B=SIN(1) results in an execution time of 12.1 seconds. So that 83 percent of the time is spent calculating sines!

Most Basic interpreters evaluate arithmetic expressions using the method of 'recursive descent'. This method has the enormous advantage that it will work directly on the arithmetic expression that was typed in, so that no pre-processing of arithmetic expressions is necessary. The method works like this: arithmetic expressions may be divided into a succession of one or more 'terms', separated by '+' or '—'. Each 'term' can consist of one or more 'factors', separated by '*' or '/' and each 'factor' may be either a function, a variable name, a number, or another expression (in brackets). You can see that this is a 'recursive' definition since a 'factor' may be an 'expression', in brackets. This sub-expression can then be thought of as dividing down in the same way, into its own 'terms' and 'factors'.

A flowchart of an expression evaluator could then be as in Figure 4. A term evaluator would be as in Figure 5 and a factor evaluator as in Figure 6. For clarity I have omitted the

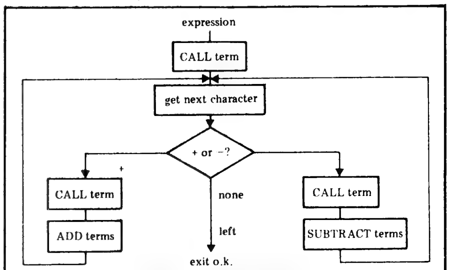


Fig 4 Expression evaluator

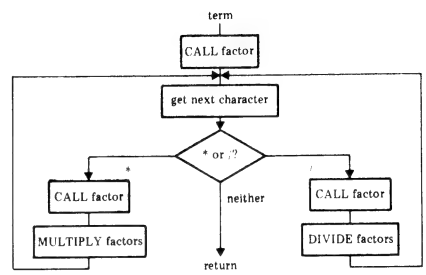


Fig 5 Term evaluator

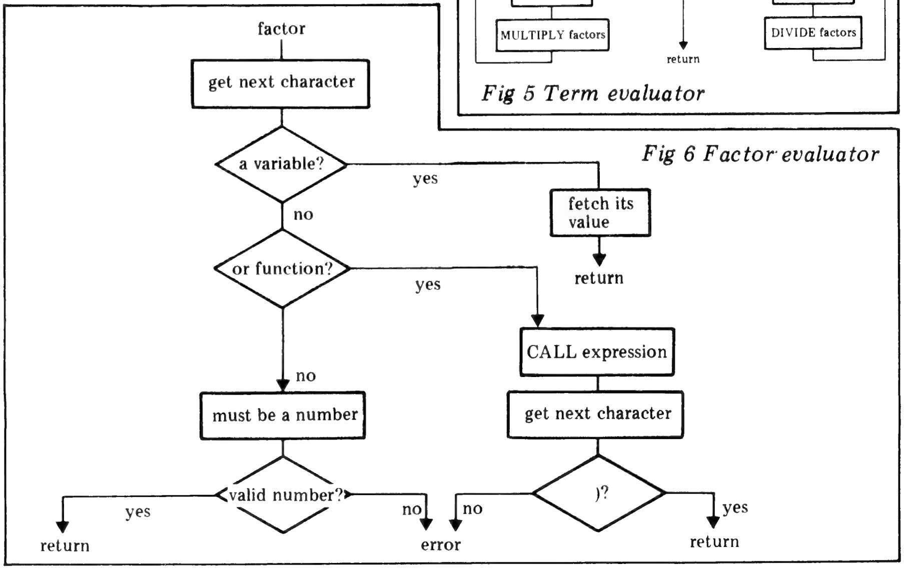


Fig 6 Factor evaluator

intermediate result pushes onto the stack, before each CALL, and pops off the stack after each return.

An interesting feature of this algorithm is that the structure of the algorithm automatically guarantees that bracketed expressions have the highest priority, * and / the next highest priority, and + and — the lowest priority, consistent with normal algebraic convention.

An alternative method of arithmetic expression evaluation — favoured in Microsoft Basics [2] is known as the 'Operator Precedence Parse'. This is more complex than 'recursive descent', and relies on each operator (+, —, *, / . . .), having a 'precedence' value associated with it. Operators with the highest priority have the highest precedence. As an arithmetic expression is scanned, whole chunks of the expression, including operators, are pushed onto a stack, until the highest precedence operator is found — parts of the expression are then removed from the stack and evaluated in reverse order of operator precedence. This method uses recursion to cope with brackets and overall the same constraints on speed apply whichever algorithm is used.

There are a number of consequences of either algorithm which affect speed of execution. The first thing to notice is that each time an open bracket occurs in the expression, the expression evaluator CALLs itself (recurses). This involves extra work and unnecessary brackets should be avoided, or expressions re-written to reduce the number of brackets.

For example, in Crystal Basic:

```
10 FOR A=1 TO 1000
20 LET B=1000+(1000*1000)
30 NEXT A
40 END
```

executes in 8.6 seconds. But replace line 20 by LET B=1000+1000*1000 (since the brackets are unnecessary here) and the execution time becomes 8.2 seconds, almost 5 percent better.

Looking further at the 'factor' routine in the method of recursive descent, if the next item in the expression is not recognised then it is assumed to be a number; numbers are, then, tested for last of all in the routine. So, if constants are pre-defined as variables, further speed improvements may result. For example:

```
 5 LET ZZZZ=1000
10 FOR A=1 TO 1000
20 LET B=ZZZZ+ZZZZ*ZZZZ
30 NEXT A
```

executes in 3.8 seconds, an astonishing 50 percent improvement over the previous example! (I have used a four-letter variable so that line 20 remains the same length, for a valid comparison — a single letter variable in fact improves the speed again — to 3.5 seconds.)

These test programs are somewhat contrived, so let me give an example of the sort of alteration to an actual program which could result in faster execution. If you have any arithmetic expressions which look like this:

LET B=10*(X+2*Y),

involving two multiplications, one addition, and a recursion for the bracketed expression, multiply out:

LET B=10*X+20*Y,

and there are still two multiplications and one addition, but no recursion. A five percent improvement already. Then try defining some constants at the start of the program —

LET A1=10 : LET A2=20

(do this right at the beginning so that these statements are executed once only), and rewrite the expression as:

LET B=A1*X+A2*Y

and this is probably significantly faster than the original expression. Of course, the program is slightly longer now, but then memory is cheap these days!

True or false?

Most Basic interpreters will allow the 'IF' statement construction:

IF variablename THEN . . . .

without any actual relational test. If you first determine how the values 'true' and 'false' are internally represented in your interpreter, you can often take advantage of this faster 'IF' statement. Try running this program on your machine:

```
 5 REM notice the multiple NEXT's,
   for speed!
10 FOR A=—5 TO 5
20 PRINT A;
30 IF A THEN PRINT "true": NEXT A
40 PRINT "false"
50 NEXT A
60 END
```

Providing your interpreter does allow the IF statement in line 30, then you may get the following result:

```
—5 true
—4 true
—3 true
—2 true
—1 true
 0 false
 1 true
 2 true
 3 true
 4 true
 5 true
```

And so, any statements that test for A not equal to 0, like

IF A< >0 THEN . . .

may simply be replaced by

IF A THEN . . .

To test if any speed improvement is achieved, run:

```
10 FOR A=1 TO 1000
20 IF A< >0 GOTO 30
30 NEXT A
```

and then replace line 20 by:

20 IF A GOTO 30

and run again. Crystal Basic shows times of 2.7 seconds, and 1.9 seconds, respectively — an improvement of 30 percent.! Of course, by the same token, 'IF A=0' could be replaced by 'IF NOT (A)' but in this case the complexity of the statement has not really been reduced — and a speed improvement is unlikely.

Interpreter v Interpreter

The classic and time-honoured means of testing and comparing Basic interpreters is by using 'Benchmark' programs. These are fine for comparison of overall systems, but tend to be misleading when what you really want to test is the 'cleverness' of a Basic interpreter. After all — you do not want the issue clouded by hardware differences like different processors, or clock speeds, or memory timing! The technique I have proposed in this article is to determine what proportion of the total RUNtime is taken up by a particular operation, by subtracting the time for an identical program without that operation. Thus, a 'standard' test for arithmetic expression evaluation might involve:

Test program 1:

```
10 FOR A=1 TO 1000
20 NEXT A
```

Test program 2:

```
10 FOR A=1 TO 1000
20 LET B=1
30 NEXT A
```

Test program 3:

```
10 FOR A=1 TO 1000
20 LET B= 'an expression'
30 NEXT A
```

The proportion of the time taken to do the 'LET' statement is

Plet = (T2—T1)/T2 * 100%

And the proportion of time to perform the arithmetic expression is

Pexpr = (T3—T2)/T3 * 100%,

where T1, T2 and T3 are the execution times of the three test programs.

These tests are not perfect, but do give a good indication of the efficacy of the arithmetic expression evaluation algorithm in an interpreter. Some examples of Plet and Pexpr for line 20:

20 LET B=1000+(1000*1000)

reveal some surprising differences between well-known interpreters, see Figure 7.

Interpreter v Compiler

Okay — so you've decided that you have made your latest Basic program go as fast as it possibly can, but it still just isn't fast enough. Should you scrap your faithful Basic interpreter and buy a new Basic or Pascal compiler? Well, the answer depends very much on what your particular program actually does. Here is an example of a Hisoft Pascal 3 program:

```
1 Program test;
2 Var  i: integer;
3      a: real;
4 Begin
5   For i: =1 to 1000 do
6     a: =1;
7 End.
```

This program executes in 0.2 seconds, a tenfold improvement over an equivalent

	T1	T2	T3	Plet	Pexpr
Crystal Basic 2.2	1.0	2.0	8.6	50%	76%
Microsoft Level 2	2.7	5.9	15.0	54%	60%
Applesoft	1.4	3.1	15.2	54%	80%
ZX81 (in fast mode)	4.4	5.9	9.7	25%	39%

Unless your computer has a timer, it is probably a good idea to replace the line 10s by: 10 FOR A=1 TO 100000, and then divide your readings by 10.

Fig 7.

Crystal Basic program. That's fine — but replace line 6 by a: sin (1) and the execution time for compiled Pascal becomes 8.4 seconds — only a 30 percent improvement over interpreted Basic! This poor improvement is explained by the fact that both the interpreter and the compiled program must do more or less the same to evaluate a sine, which is taking up most of the time anyway. Thus compilers do give significant speed improvements if only simple arithmetic is involved — but these worsen as soon as floating point arithmetic is introduced.

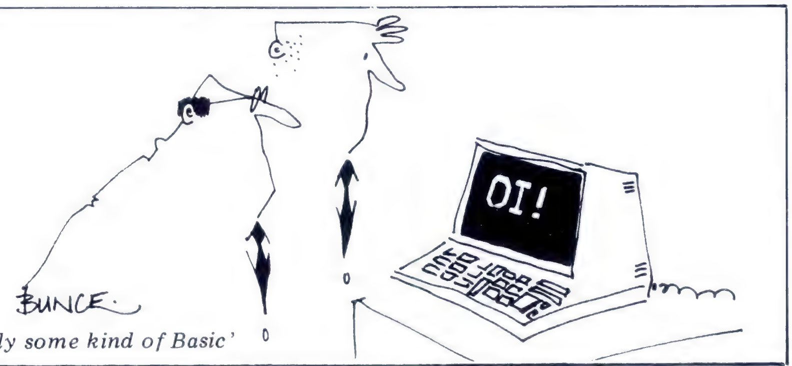
Conclusions

I hope that I have shown that it is worth getting to know your favourite interpreter — using the two techniques of examining the stored program and running test programs like the examples I have used (or studying the assembler source if you are lucky enough to have access to it). That knowledge can then be used to write more efficient, and faster programs. The techniques I have proposed are by no means definitive, or complete, and I would welcome any new ideas, or revelations, on this subject.

I must thank my long suffering colleagues, who patiently waited while I attacked their machines with a stop-watch!

References

[1] Greeb, FJ, 'Denver Tiny Basic for 8080s', *Dr Dobbs*, March 1976.
[2] Farvour, J, 'Microsoft Basic Decoded, and other mysteries', *TRS80 Information Series*, Vol. 2.



'Obviously some kind of Basic'

ZX81 PRINTER

Maggie Burton tests the cheapest printer on the market.

The ZX81 (and its predecessor, the ZX80) always lacked one vital peripheral — a printer. Programs have had to be typed or handwritten and the only way of keeping a convenient record was to use a cassette recorder. Now, after having been heralded ever since the ZX81 launch, the printer is available to complete the system.

It is an electrostatic printer — that is, it prints on metallic (aluminium-coated) paper using conductive styli to evaporate the metal coating where a dot is wanted, thereby revealing a black layer in the appropriate places underneath. However, unlike other electrostatic printers, which have a whole row of styli, the Sinclair only has two, one on either side of a revolving belt. One stylus prints a line of dots from left to right, while the other returns to start the next one.

By normal standards, it doesn't print at all fast — about 21.5 cps, in fact — but for what it has to do this is quite adequate. I thought, until I actually timed it, that it was fast but that's how it looks when it's working. Obviously the graphics characters are printed more slowly than the normal ones as they have more dots in them. This is demonstrated effectively by a program in the manual which prints out the whole character set. All the graphics are printed out in succession so the printer slows down considerably while it is printing them. The cps timing I mentioned just now was the average timing for five runs of this program (and a few nifty calculations to boot).

It reacts almost instantaneously to a COPY or LLIST command from the computer. You can print out anything with it that will display on the screen and it will keep going almost indefinitely without any apparent ill effects apart from a rather strange smell if you put your nose close enough to the print head. Somehow, I can't see it being used for word processing or some other such application but a printout of a pretty graphics program would certainly not be out of place on some young computerist's bedroom wall. It is also useful from our point of view because, hopefully, from now on we won't be wading our way through reams and reams of untidy handwritten ZX81 listings! It certainly makes it a lot easier to keep an organised record of what you last did with the computer when you had to stop half way through doing something. All you have to do is press COPY (NEWLINE) and it will record everything from the TV screen for future reference. LLIST will list a program onto the printer instead of the screen. The paper will photocopy well so you can send copies of your favourite programs to your friends.

Now on to maintenance The manual says you have to clean the print head from time to time as it tends to get clogged up with the black stuff from underneath the metal on the paper. This is done with the reel carrier out and is not a difficult undertaking, although it should be done with a soft tissue or brush and without using any chemicals or hard objects. Apart from cleaning, it doesn't need any maintenance at all so it isn't exactly demanding to look after. Changing a reel of paper is easy as well: the whole reel carrier slides out when you press it underneath and the reel simply clips into it. Paper sells for around $30.00 for five rolls. No other paper is suitable — I shouldn't think it would be possible to find another make that would fit the bill in any case. As a last warning, I wouldn't take it to bits unless you really know what you're doing because you'll have terrible trouble putting it back together again, as our editor found out when he had finished looking inside it!

After using it for a while, I felt myself sufficiently acquainted with it to make a few complaints, all of which are relatively small. The first is the amount of exposed wiring. The connector is open at the side so you can see the pretty coloured wiring in it. I'm sure this isn't an overwhelmingly serious fault but it looks shoddy and if you spilt your coffee on it while computing it wouldn't do it much good. The connector, by the way, slots in where the RAMpack goes and has a socket incorporated at the back of it for the RAMpack to plug in as well. This leads me to my next complaint. As any ZX81 owner or user will probably know, the RAMpack is a bit stroppy and may well crash the whole system if it touches anything apart from the computer. Connecting it to the back of the printer makes it stick out further and is more liable to surreptitious nudges from stray objects. This caused a great deal of swearing and cursing when I was using the system, but it can be overcome by putting the ZX81 on a pile of books so that the RAMpack doesn't droop and touch the table as it is inclined to do when the printer is being used. Really, though, I don't think Sinclair could have found another convenient place to join it all together apart from, perhaps, another socket at the other side, but then the system would still be just as likely to crash. Finally, I found the length of cable between a printer and ZX81 a bit short, which means they are rather too close together. Although this doesn't prevent you from doing anything that a long cable would promote, it can be rather annoying at times.

Still, brickbats delivered, I do think it's good value and, at the price, it's totally unique. The Sinclair costs $190.00 which includes a beffier PSU to power computer and printer and replaces the original one. I can't see anyone having the slightest trouble setting it up and getting it working — it really is child's play and I'm sure many kids (and dads) will get many happy hours of LLISTin and COPYing from it. I wonder how many hardware freaks will find ways of hooking it up to their programmable calculators . . . ?

NEWCOMERS-START HERE

What follows is a brief guide for the microcomputing novice. It has been designed for quick reference — with all the key words in bold type; of course if you're feeling adventurous, you're welcome to read it right through. Whichever way, we trust you will find it helpful. Happy Microcomputing!

Welcome to the confusing world of the microcomputer. First of all, don't be fooled; there's nothing complicated about this business, it's just that we're surrounded by an immense amount of necessary jargon. Imagine if we had to continually say "numbering system with a radix of sixteen in which the letters A to F represent the values 10 to 15", when instead we can simply say "hex". No doubt soon many of the words and phrases we are about to explain will eventually fall into common English usage. Until that time, APC will be publishing this guide – every month.

We'll start by considering a microcomputer's functions and then examine the physical components necessary to implement these functions.

The microcomputer is capable of receiving information, processing it, storing the results or sending them somewhere else. All this information is called **data** and it comprises numbers, letters and special symbols which can be read by humans. Although the data are (yes, it's plural) accepted and output by the computer in 'human' form, inside it's a different story – they must be held in the form of an electronic code. This code is called **binary** – a system of numbering which uses only Os and 1s. Thus in most micros each character, number or symbol is represented by eight binary digits or **bits** as they are called, ranging from 00000000 to 11111111.
To simplify communication between computers, several standard coding systems exist, the most common being **ASCII** (American Standard Code for Information Interchange). As an example of this standard, the number five is represented as 00110101 – complicated for humans, but easy for the computer! This collection of eight bits is called a byte and computer freaks who spend a lot of time messing around with bits and bytes use a half way human representation called **hex.** The hex equivalent of a byte is obtained by giving each half a single character code (0–9, A–F):
0=0000, 1=0001, 2=0010, 3=0011, 4=0100, 5=0101 ...... E=1110 and F=1111.
Our example of 5 is therefore 35 in hex. This makes it easier for humans to handle complicated collections of Os and 1s. The machine detects these Os and 1s by recognising different voltage levels.

The computer processes data by reshuffling, performing arithmetic on, or by comparing them with other data. It's the latter function that gives a computer its apparent 'intelligence' – the ability to make decisions and to act upon them. It has to be given a set of rules in order to do this and, once again, these rules are stored in memory as bytes. The rules are called **programs** and while they can be input in binary or hex (**machine code** programming), the usual method is to have a special program which translates English or near-English into machine code. This speeds programming considerably; the nearer the **programming language** is to English, the faster the programming time. On the other hand, program execution speed tends to be slower.

The most common microcomputer language is **BASIC.** Program instructions are typed in at the keyboard, to be coded and stored in the computer's memory. To run such a program the computer uses an **interpreter** which picks up each English-type **instruction,** translates it into machine code and then feeds it into the **processor** for execution. It has to do this each time the same instruction has to be executed.

Two strange words you will hear in connection with BASIC are **PEEK** and **POKE.** They give the programmer access to the memory of the machine. It is possible to read (PEEK) the contents of a byte in the computer and to modify a byte (POKE).

Moving on to hardware, this means the physical components of a computer system as opposed to software – the programs needed to make the system work.

At the heart of the microcomputer system is the central processing unit **(CPU)**, a single microprocessor chip with supporting devices such as **buffers**, which 'amplify' they CPU's signals for use by other components in the system. The packaged chips are either soldered directly to a printed circuit board **(PCB)** or are mounted in sockets.

In some microcomputers, the entire system is mounted on a single, large, PCB; in others a bus system is used, comprising a long PCB holding a number of interconnected sockets. Plugged into these are several smaller PCBs, each with a specific function – for instance, one card would hold the CPU and its support chips. The most widely-used bus system is called the **S100**.

The CPU needs **memory** in which to keep the programs and data. Microcomputers generally two types of memory, **RAM** (Random Access Memory) and **ROM** (Read Only Memory). The CPU can read information stored in RAM – and also put information into RAM. Two types of RAM exist – **static** and **dynamic**; all you really need to know is that dynamic RAM uses less power and is less expensive than static, but it requires, additional, complex, circuitry to make it work. Both types of RAM lose their contents when power is switched off, whereas ROM retains its contents permanently. Not surprisingly, manufacturers often store interpreters and the like in ROM. The CPU can only read the ROM's contents and cannot alter them in any way. You can buy special ROMs called **PROMs** (Programmable ROMs) and **EPROMs** (Erasable PROMs) which can programmed using a special device; EPROMs can be erased using ultra-violet light.

Because RAM loses its contents when power is switched off, **cassettes** and **floppy discs** are used to save programs and data for later use. Audio-type tape recorders are often used by converting data to a series of audio tones and recording them; later the computer can listen to these same tones and re-convert them into data. Various methods are used for this, so a cassette recorded by one make of computer won't necessarily work on another make. It takes a long time to record and playback information and it's difficult to locate one specific item among a whole mass of information on a cassette; therefore, to overcome these problems, floppy discs are used on more sophisticated systems.

A floppy disc is made of thin plastic, coated with a magnetic recording surface rather like that used on tape. The disc, in its protective envelope is placed in a disc drive which rotates it and moves a **read/write head** across the disc's surface. The disc is divided into concentric rings called **tracks**, each of which is in turn subdivided into sectors. Using a program called a **disc operating system,** the computer keeps track of exactly where information is on the disc and it can get to any item of data by moving the head to the appropriate track and them waiting for the right sector to come round. Two methods are used to tell the computer where on a track each sector starts; **soft sectoring** where special signals are recorded on the surface and **hard sectoring** where holes are punched through the disc and around the central hole, one per sector.

Half-way between cassettes and discs is the **stringy floppy** – a miniature continuous loop tape cartridge, faster than a cassette but cheaper than a disc system. Hard disc systems are also available for microcomputers; they store more information than floppy discs, are more reliable and information can be transferred to and from them much more quickly.

You, the user, must be able to communicate with the computer and the generally accepted minimum for this is the visual display unit **(VDU)**, which looks like a TV screen with a typewriter-style keyboard; sometimes these are built into the system, sometimes they're separate. If you want a written record **(hard copy)** of the computers output, you'll need a **printer**.

The computer can send out and receive information in two forms – **parallel** and **serial**. Parallel input/output (I/O) requires a series of wires to connect the computer to another device, such as a printer, and it sends out data a byte at a time, a separate wire carrying each bit. Serial I/O involves sending data one bit at a time along a single piece of wire, with extra bits added to tell the receiving device when a byte is about to start and when it has finished. The speed that data is transmitted is referred to as **baud rate** and, very roughly, the baud rate divided by 10 equals the number of bytes being sent per second.

To ensure that both receiver and transmitter link up without any electrical horrors, standards exist for serial interfaces the most common is **RS232** (or **V24**) while, for parallel interfaces to printers, the **Centronics** standard is popular.

Finally, a **modem** connects a computer, via a serial interface to the telephone system allowing two computers with modems to exchange information. A modem must be wired into the telephone system and you need Telecom's permission; instead you could use an **acoustic coupler**, which has two obscene-looking rubber cups into which the handset fits, and which has no electrical connection with the phone system – Telecom isn't so uppity about the use of these.

FRAMES OF REFERENCE:

A DP MANAGER'S GUIDE TO MICROS

Many series have been written to introduce microcomputers to the new computer user; some are valuable to computer people as guides to micro computing, but none are addressed to the computer professional's dilemma: how to manage in the micro age when technology is developing at a bewildering pace, microcomputers are sprouting everywhere in user departments and no one is quite sure where micros stop and mainframes start. For the last four years Alan Wood, who is associated with four microtechnology companies and came to micros from a traditional DP background, has been installing micro systems and advising large firms on micros. This series reflects his own continuous adjustment to micro shock and the practical applications of microtechnology within the existing computing framework of the large organisation. It has been written expressly for the heads of management services, computer managers and computer professionals in user and supplier companies. Its object is to provide a framework in which the rapid developments in microtechnology can be placed and viewed in the context of other computing. It provides pegs on which the professional can hang information and it includes guidelines for the application and control of microcomputing. Such is the pace of development that those responsible for establishing strategy face the necessity of a continuing review to assimilate the impact of the new technology on their plans. The series provides a useful start to the process of assimilation and, for those who have already begun, a valuable second opinion to confirm and sometimes question their views.

PART 1: MASTERING THE NEW COMPUTING

The last decades of the 20th century will be remembered as an era of new departures. The concentration on reducing our calorific intake has created the new cuisine; the search for alternative energy sources is harnessing the sun and the sea; interest in the paranormal may be heralding a new science; the polarisation of political parties is creating new politics; the departure of the war generation is breathing fresh air into business and industry; and the arrival of the microchip is the start of new computing.

By the end of the 80s, computers will be as commonplace as calculators were at the end of the 70s. By the end of the 90s, we will be living in a computerised video society. The combining forces of computing and communications technology, the development and sophistication of existing channels — television, telephone, telecoms, video and microtechnology — are inexorably moving us towards a 21st century lifestyle previously regarded as science fiction. These changes mean radical alterations in our lives and, particularly, in our work. Everyone will have to go back to school, and keep going back to school, to keep pace. But the school will be in our homes and offices in the guise of a terminal on which we play videograms for passive learning and programs for participative learning.

Computer professionals will have to learn a new trade and keep on learning to stay in touch with, and find a place in, a technology-led revolution. In old computing, you could learn your ICL 1900 and IBM 360 and be sure they would be around for ten or more years. With some refreshing seminars it was relatively easy to keep pace. The arrival of minicomputers from DEC, Data General, etc, hotted up the pace and some DP people took a long time to adjust to interactive computing. But the pace of mini development is positively pedestrian compared to micros. The micro industry only started in 1975, yet there are now more than two million machines installed worldwide. Apple began in a garage in 1977 and has sold 300,000 machines. The micros of two years ago have already been superseded by second generation micros: the micros of today will be superseded by a third generation in another two years.

When micros started, there was no software available. Now there are thousands of packages, several hundred of which are technically excellent and only cost hundreds of dollars. The cost of developing some systems on micros has been cut by 75 per cent in two years with the arrival of development aids. And we have achieved more portability of software and skills in the micro industry in a few years than has ever been achieved even within a traditional computer manufacturer's own line. The computer professionals who ignore these trends are heading for a future dole queue; the DP manager who turns away is on the road to early retirement.

In August last year, IBM announced its Personal Computer and, overnight, micros were not just fashionable but respectable. Apple Computer, the company that had rocketed to a billion-dollar stock exchange valuation in just four years, welcomed the announcement by taking a full page advertisement in the *Wall Street Journal*. DP managers, eager that micros were a passing fad or that IBM would come up with an answer to the upstarts, had one of their wishes fulfilled. The burgeoning software industry got its biggest lift since unbundling with the information that the IBM machine would come with industry classics — Microsoft Basic, Personal Software's Visicalc and the Peachtree accounting packages. The market for hardware and software took a quantum leap with the promise that IBM would put its Personal Computers everywhere there were electronic typewriters and terminals — and a few other places besides.

Dramatic though the impact of the IBM announcement was — the sales equivalent of the Pope endorsing birth control products — it only set the seal of approval on an industry already well developed with missionary zeal by acolytes of such strange gods as Tandy, PET, CP/M and the S100 bus. Moreover, far from answering the DP manager's prayer to conformity, IBM's entry into micros is one more development which will further fragment, if not shatter, the Holy DP Empire as computers become as easy to use and as accessible as cars. We are only a small step away from taking 50 megabyte video discs and video input cameras right out of 'Tomorrow's World' and into our offices, thereby decimating two of the largest remaining costs — bulk storage and file creation — that are a deterrent to mass computerisation.

The DPM's micro survival kit

Acquire:	a personal computer, e.g. Apple
Select:	some packages: Visicalc or Wordstar
Appoint:	a Micro Project Manager in your department
Subscribe to:	*Byte, Australian Personal Computer,*
Read:	*The Mighty Micro* (for stimulation), Osborne series (for detail), Zak's *Your First Computer* (simple introduction), this series (practical advice)
Retain:	a microsystems company as advisor and supplier
Maintain:	register of all micros and applications in your company
Implement:	pilot stand-alone and comms-to-mainframe projects
Run:	courses on micros for DP department staff and users
Revise:	computer thinking and five year plan to incorporate new technology

and the processing power of a PDP 11/44.

— *Fact 3:* At one half the cost of the conventional mini, you can replace a 24-station shared processor minicomputer with a 24-station local area micro network and give every user his own processor.

Some DP people have recently embraced micros with all the enthusiasm and blinkered vision of new converts, neither looking to the left of them at the weaknesses in the current machines, nor to the right of them where they would find first generation micros being superseded by second generation machines. Others have stood back and watched their users get on with it, waited, and sometimes prayed for them to fall into the snakepit of undersizing, inadequate software and hardware breakdowns.

Once some pegs have been established on which to hang and relate information, the transition to new technology is achievable for those who want to make it. Apart from their physical size and the software available, 8-bit microcomputers have similar power and memory size to those old 1401s, although they are used in interactive rather than batch mode. Working in micros is also like very old computing, providing closeness of the systems worker to the machine and to the user. Most micro systems are implemented by one-man teams working directly for a user and employing prototyping techniques. Few micro projects need more than three or four people, or a timescale greater than six months. Micros lend themselves to rapid interactive development (compute time on micros is amazingly cheap compared to mainframes) and the use of development aids to prototype systems. You can easily show the user what his screen will look like before you implement his systems. A partnership with the user is formed during the development stage and the systems engineer, who designs, programs and uses software tools, also trains the user to run his systems after completion. The user will typically go on to use some of the tools, report writers, sorts, etc, to access his system. In micros we were reversing the trends we started 15 years and more ago. We are giving users back their filing cabinets — but automating them first.

The DPM's micro survival kit

The first law of survival is education. Read some books on micros, such as *The Mighty Micro* and *Your First Computer.* Obtain magazines regularly, e.g. *Australian Personal Computer, Electronics, Interface Age* (the last two are American). Attend courses on micros, preferably of the workshop variety. Obtain a personal computer and either use it in your office or, until the embarrassment has worn off, at home. With your personal computer, say an Apple, an Osborne or a Superbrain, acquire some packages. A good shopping list will include word processing (Wordstar, Magic Wand), a rows-and-columns product (Visicalc, SuperCalc), an information manager (Selector IV, Datastar). You should also include in your repertoire the industry-standard languages, Microsoft Basic and Micro Focus CIS Cobol. The next sensible action will be to appoint one of your bright young staff to take a special interest in micros. If you are a manager in a large company, a micro group with a team of people will be needed. Then you should find a microsystems company and professional dealer from whom you can obtain sensible advice, buy equipment software, training and support services. The best dealer companies operate as surrogate micro departments and their resources are drawn on as and when needed.

Micro strength

Whatever we say about micro limitations today becomes quickly dated. The 32-bit Mainframe Micro has already been announced by Intel. The 16-bit Mini-Micro is with us. In 1980 the norm was a 48k RAM single user, 8-bit micro. In 1981 the four-user, 256k RAM and 10 megabyte micro became commonplace. In 1982, the powerful 16-bit machines put a megabyte of memory on the desktop. These will be followed by revolutions in storage, most exciting of which, the video disk, could provide offices with storage for as little as $2 per megabyte!

The outstanding advantages of today's micros can be summarised as: (a) low cost, (b) rapid delivery, (c) minute size and (d) software availability.

The low cost of micro systems is not due to the fall in processor prices alone but also comes from the collapse in peripheral prices, brought about by the microchip itself and the massively increased sales it has engendered. In the space of two years, some printer prices have fallen to one third their former level and visual display screen prices have been chopped in half. You can buy an 80-cps, 132-columns printer for $1000 and a VDU for as little as $670. Nor are these shoddy goods: NEC, Ricoh and Epson printers have the traditional Japanese reputation for reliability. (It is significant in this regard that many American suppliers are badge engineering Japanese products as part of their own offerings!).

The second virtue of the micro is its ready availability. The waiting times associated with mainframes or even minis simply do not apply in the micro industry. The small personal computers, Apple, PET, Tandy, Sharp, are instantly available. The more powerful business machines are delivered typically ten to 30 days from order. You could put in a local area network to replace a mini in 30 days and some companies have done so. This rapid delivery really does encourage the use of microcomputers and assists management to make things happen quickly. Users are universally fed up with the tediously long time it takes their computer departments to deliver systems.

The third virtue of the micro is its minute size. When you can literally put a computer on your desk, the saving in space is significant. And with office rentals ever on the increase, the cost of the space is no small consideration. Micros will continue to produce greater power in smaller spaces. The practical limitation is the size we humans need to read in comfort and, until voice entry becomes the norm, keyboards have to fit fingers.

The fourth virtue of the microcomputer is the software library available to achieve practical working systems in a very short space of time. Most micros are used with existing packages and development aids, and the number and sophistication of software products is growing daily. The low cost of micros

Things I wish I had known five years ago

1. CP/M (Control Program Microprocessor) would become the de facto industry standard micro operating system and sell more than 250,000 copies.
2. Apple Computer would sell more than 300,000 machines and be valued at over one billion dollars on the New York Stock Exchange.
3. Peripheral prices would quarter as new technology cut costs and volume sales when new markets took hold.
4. Systems Software Manufacturers would become the norm for the supply of operating systems and languages, eg, Micro Focus, Microsoft, Digital Research.
5. Software Publishers would sell packages in tens of thousands, eg, Lifeboat, Personal Software.
6. Chip prices would fall from $16 for 4k dynamic RAMs to $2 for 16k dynamic RAMs; from $80 for an 8080A processor to $5 for a Z80A processor.

HERE NOW!

sinclair

ZX PRINTER

Designed exclusively for use with the ZX81 (ZX80 with 8K BASIC ROM), the printer offers a full alphanumerics across 32 columns, *and* highly sophisticated graphics. Special features include COPY which prints out exactly what is on the whole TV screen without need for further instruction.

AMAZING LOW PRICE $190 incl. Sales Tax

sinclair COMPUTER PRODUCTS

Available from the following leading stores —

QUEENSLAND:
MYER – ALL BRANCHES
SOFTWARE 80, Shop 11/200 Mogill Rd., Taringa, 4068 (07) 371 6996
DATACOM, Shop 16, S.G.I.O. Arcade, Bundaberg, 4670 (071) 71 4740

A.C.T.:
COMPUTERWORLD, Shop G71, The Bridge, Woden Plaza (062) 81 1368

TASMANIA:
J. WALSH & SONS PTY LTD, 130 Macquarie St, Hobart, 7000 (002) 34 7511
BIRCHALLS, 118-120 Brisbane St. Launceston, 2250 (003) 31 3011

VICTORIA:
MYER MELBOURNE, City Store Only
RADIO PARTS GROUP, 562 Spencer St. West Melbourne, 3000 (03) 329 7888
ROD IRVING ELECTRONICS, 425 High St, Northcote, 3070 (03) 489 8131
MINIT COMPUTER SERVICE, 119 McCrae St, Bendigo, 3550 (054) 43 2589
B.B.J. COMPUTER SHOP PTY LTD, 88 Albert Rd., South Melbourne (03) 699 5622

W.A.:
MICRO BASE, 127 Fitzgerald St., Perth, 6000 (09) 329 9308

S.A.:
MYER ADELAIDE
ACUIS TRADING, 185 Pirie St., Adelaide, 5000 (08) 223 1900

N.S.W.:
B.B.J. COMPUTER SHOP PTY LTD 329 Pacific Hwy, Crows Nest, 2065 (02) 922 4022
CITY PERSONAL COMPUTER, 75 Castlereagh St, Sydney, 2000 (02) 233 8992
DAVID REID ELECTRONICS PTY LTD, 127 York St, Sydney, 2000 (02) 29 6601
COMPUTERWAVE PTY LTD, Lwr Ground Floor, Myer, Sydney, 2000 (02) 238 9111
COMPUTERWAVE PTY LTD, Albury (060) 21 5933
TRINITY COMPUTING, Shop 5, 1/9 Palmer St, Parramatta, 2150 (02) 683 4349
CISA MICROCOMPUTERS PTY LTD Mezzanine Floor 89 York St Sydney (02) 29 1599
DIRECT COMPUTER SALES, 198 Forest Rd, Hurstville, 2220 (02) 570 8344
DICK SMITH ELECTRONICS – All Stores
COMPUTER GALLERIE, 66 Walker St, North Sydney 2060 (02) 929 5497
AED MICROCOMPUTER PRODUCTS, 130 Military Rd, Guildford, 2161 (02) 632 6301
COMPUTERS GALORE, 99 Military Rd, Neutral Bay, 2089 (02) 908 2355

CMC22

ORDER FORM: SINCLAIR EQUIPMENT (A'SIA) P/L
86-88 Nicholson St., Abbottsford, Vic. 3067. Ph. (03) 419 3033.

QTY	ITEM	ITEM PRICE	TOTAL
	Ready assembled ZX81 Sinclair Personal Computer including mains adaptor, leads, BASIC manual	$250	
	16K-BYTE RAM pack	$150	
	8K-ROM (only required for ZX80)	$ 75	
	ZX Printer	$190	
	1.2 Amp Adaptor for use with ZX Printer	$ 17.50	

I enclose cheque/Bankcard/Diners Club/Amex **Total** ______

Name ______
Address ______ P'code ______
Signature ______ APC

also means you can use them in dedicated development, an approach which improves programmer productivity. Some DP departments have found that it is cheaper and quicker to employ micros for Cobol program development and subsequently transfer the programs to their mainframes.

Software gateways to the future

Microsoft Basic:	Available on: 8-bit 8080, Z80, 6502; 16-bit 8086, M68000, Z8000 Under: CP/M, MP/M, Xenix (Unix)
Micro Focus CIS Cobol:	Available on: 8080, Z80, LSI II, 8086, CP/M, MP/M, Unix, etc.
Digital Research:	CP/M single user operating system; MP/M multi-user operating system.
Bell Labs:	Unix, C and derivatives: multi user development and operating environment
UCSD Pascal:	For portable software packages.

Micro weakness

What are micros not good for? Presently, the practical disk storage limitation of the microcomputer is about 100 megabytes. Micros are generally not suited to large file handling and manipulation. The software is not available for such applications, nor is suitable hardware. Neither are micros suited to the bigger multi-user applications. The operating systems on micros are not yet good enough to handle dozens of terminals with high transaction volumes. However, the low cost of micros has led to a completely different solution to the multi-user need: the local area network. On a local area network each user gets his own processor and shares the central disks and line printers. Local area networks are still limited in their disk capacities — around 200 megabytes — but these boundaries will soon be extended. A local area network is a real alternative to the multi-user mini, being both cheaper and more resilient.

The main weaknesses associated with micros are not technical. The greatest problem is that the proliferation of micros in larger companies is likely to lead to a liquorice allsorts of machines and incompatible systems. The proliferation can also produce a terrible waste in expensive man time as new users re-invent the wheel. Moreover, the tendency to put micros in the hands of undisciplined users can create embarrassing situations when a member of staff leaves a company and he or she is the only person who knows the system. It is essential that companies have a planned strategy for microcomputing to minimise waste, insure against expensive errors and get the best out of the new technology.

Strategy for microcomputing

Medium and larger-sized organisations should establish a strategy for microcomputing which sets the new technology in the context of what is already in their companies. Such a strategy will include policy guidelines which are revised annually. It will consider in depth and detail the positioning of microcomputing in relation to mainframes, minis, word processing, timesharing and communications. It will express standards for microcomputing with special reference to software. It will insist on a fast response to user needs and a mechanism that produces a fast response, without which disillusioned users will find their own way to microcomputerise. It will establish criteria for justification and payback on microsystems. (Micros are typically written off over three years and show a payback in not more than two years.) It will contain an action programme to educate DP staff and users to implement pilot systems.

The software gateway

DP has been consumed for many years with applying hardware standards or standardising on particular manufacturers' equipment. In microcomputing, it is far more important to establish software standards, and to use existing tools and packages, than it is to establish hardware standards. It is likely that the hardware you will be taking on in three years time will bear little relationship, except in the software which runs on it, to that which you have today. In implementing systems on micros, it is important to start with the user requirements and not with the hardware you have. It is not a case of how one can implement this system on one's ICL or IBM machine under such and such operating systems and language; rather, it is very much a case of finding the shortest software route to solve the problem. What package fits the bill? Or what development aids are available to speed the implementation? And if you are using programming languages, then it should be either Microsoft Basic, CIS Cobol or Pascal, to provide you with a gateway to future technology. These are the industry-standard languages and if you use others you do so at your own peril. Not only will you be investing time in learning, but you may be limiting your future options and tying yourself into obsolete technology. For the first time in the history of computing, computers are being designed to fit the software available not vice versa. The semiconductor manufacturers, Intel, Zilog and Motorola, have adopted Unix both to get their 16-bit technology to market faster and to provide a route to their 32-bit offerings. CP/M, the single user operating system from Digital Research, has sold in millions and is the industry standard. Its multi-user brother, MP/M, is being adopted by many of the commercial manufacturers both for 8-bit and 16-bit multi-user applications. There are other operating systems, including the manufacturers' own. But all the independent operating systems worth using have also got Microsoft Basic and CIS Cobol on them. The IBM machine comes with Microsoft Basic; CIS Cobol is generally available on MP/M and Unix. These languages provide a genuine software gateway to the future.

Hardware standards

The most common processor for hobby computers is the Mostek 6502, used in the Apple, PET and Atari. All these machines have their own languages and operating systems. Software written on one is not readily portable to the others. They are at their best when used for education, numeric applications, low volume files and word processing. They are most often employed by using existing packages and tools.

By far the most popular processor for business microcomputers is the Zilog Z80. This contains the instruction set of the Intel 8080 and is compatible with it. CP/M runs on the Z80 and 8080, providing access to a vast range of software. Z80 machines have commonly been adopted as the standard to provide portability of expertise and software in the larger company.

The Z80 computer comes in two implementations: the S100 bus version and the 'own bus' version. The bus is simply the internal communications system of the computer. The S100 bus has

100 common communications or connecting lines. It provides the facility to plug in additional hardware cards, much as you can plug in three-pin plugs. Some microcomputer suppliers have developed their own bus structures, eg, Altos, Intertec, and Zilog. A greater number have adopted the S100 bus, which has now been taken up by the IEEE and become an international hardware standard. Manufacturers using the S100 bus include Dynabyte, Cromemco, Micromation, North Star, Industrial Micro Systems, Comart, Vector Graphic. Single or own bus machines have a reputation for greater reliability and are simpler, but it is more difficult to change their configuration. S100-bus machines are easy to re-configure and you can add or change their internal boards very readily. They also have access to an ever-growing variety of additional hardware: colour graphics, viewdata, emulation and other plug-in boards. Some newer implementations of the S100 bus are more reliable than the older machines.

So far, we have been talking about 8-bit processors, processors that will continue to be useful for some time to come. The newer 16-bit processors are just coming into the market place and it isn't yet clear which will become the industry standards as the Z80 became the 8-bit standard. It seems likely, however, that the Intel 8086 and Intel lookalikes are going to capture the early volume market. Several of the existing suppliers, Altos, Micro V and Vector, have opted for Intel. Cromemco has switched to the Motorola 68000 and so, it is rumoured, has Apple. It seems probable that Intel will get the lion's share of the market and Motorola seems set to pick up the more sophisticated, but less voluminous, portion. Zilog is presently showing third, with its own computer and most notable Z8000 machine available.

The profusion of processors, both 8- and 16-bit, emphasises the need for managers to establish software standards. All these processors can be programmed with the languages already mentioned. If you want rapid implementation of an interactive system, Microsoft Basic is a good choice. If you want good file handling and processing, as well as maintainable code, CIS Cobol will suit for now and the future. If you're building software tools and packages, Pascal is increasingly favoured.

You will also be well advised to limit the choice of peripherals since it becomes very difficult to support hundreds of different devices. Of the newer manufacturers, Anadex, Epson, Ricoh and NEC have made a mark in printers, alongside Diablo, Texas Instruments and Centronics. The VDU market is highly competitive, with Lear Siegler, Teleray, Televideo and now even IBM, all slugging it out. The most important word in peripherals is reliability: do not go for the cheapest or the latest, for this reason alone. Ask the supplier and heed his advice when he tells you what works day-in and day-out.

First commandment of microcomputing

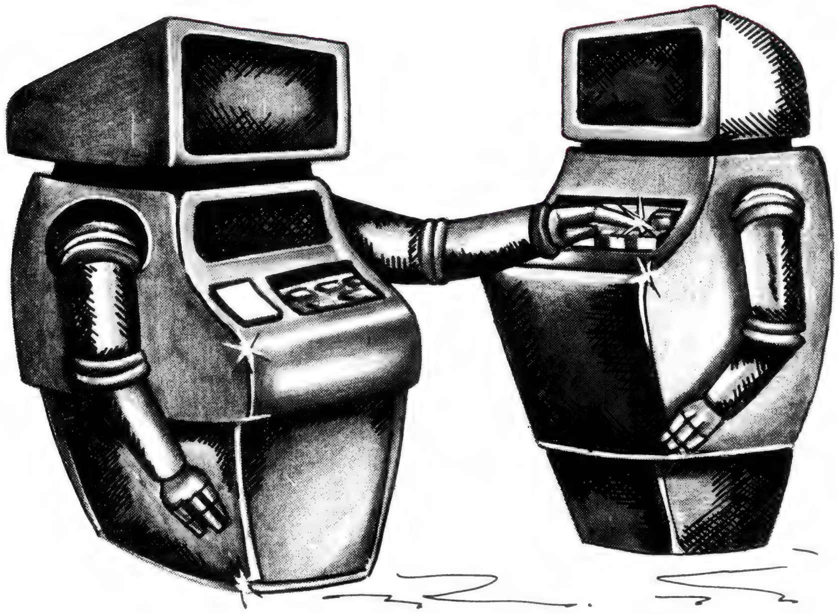
Thus far we have provided some comfort to the computer professional with familiar references to software and hardware standards. These points are valid as a starting ground to understanding and to planning. But they are only an introduction to the first commandment of microcomputing: *Thou shalt not treat a microcomputer as a computer.* Thou shalt treat it as an application device.

Word processors are not microcomputers, they are application machines. Hi-fis are not electronic consumables; they are music players. Microcomputers are not computers. . . they are stock machines, budgeting machines, mailing machines. When you are considering micros, you look at the languages, packages and tools first, last and always. Then you decide on which reliable micro you are going to play them.

If you want a membership record and subscription system, with mailing and word processing, you could hand-program it for $20,000 using Basic, or you could build it with available programming tools for around $5000. And you could use those tools for other applications, too. Aids of this class are literally slashing as much as 80 per cent off traditional development costs for small systems.

END

HOW COMPUTERS COMMUNICATE.



PART IV

THE STANDARD INTERFACE

by Hewlett Packard's Steve Leibson

Computer designers constantly strive to implement the latest parts and the fastest logic in new and different configurations. This characteristic has created a volatile situation in the computer industry. Additionally, designers of peripherals to be connected to the computers are creating entirely new classes of devices.

The end result is a multitude of interfaces, all of which have been optimized for the instrument to which they belong. But very few of them are compatible with each other.

ADOPTING A STANDARD

The situation is similar to the American railroads of the early 1800s. Dozens of track gauges existed and cars of one railroad could not travel on the tracks of another. Just as the railroads quickly standardized the track gauges, the computer industry agreed on an interfacing standard that the Institute of Electrical and Electronic Engineers (IEEE) subsequently published in 1975.

It was the first comprehensive, nearly universal interfacing standard for computers and instrumentation. That first version, IEEE Standard Digital Interface for Programmable Instrumentation, or IEEE Std 488-1975, was revised in 1978, and now is called IEEE Std 488-1978.

This standard defined a general-purpose interface, designed for instrumentation systems requiring limited-distance communications. The intent of IEEE 488-1978 is to pin down as many variables in an interface as possible without defining the actual use of the interface.

In addition, the interface is defined without reference to the hardware circuitry required to implement the interface. This allows newer products to take advantage of newer technologies, permitting faster, less expensive construction of devices and systems.

In the previous article in this series, we discussed the parallel interface. That interface had 16 input and 16 output lines, to make possible interfacing to as many different devices as possible. A very popular version of the parallel interface has no connector at the end of the cable. The system builder must add the appropriate connector for his peripheral, since there is no standard for either the connectors or how the pins in the connector are to be used.

Connector and pin usage are precisely specified in IEEE standard 488, as are signal levels, both voltage and current, and signal timings. Thus a system becomes a "remove-from-the-box-and-plug-together" operation. The hardware of interconnection is defined so that two interconnected instruments can communicate, although understanding is not guaranteed by the standard.

CONTROLLING, TALKING AND LISTENING

On the I/O bus, only two entities reside; the computer and the interface. The computer always controls the I/O bus, and the interfaces react to commands from the computer.

Three types of devices exist on the interface. These entities are actually attributes, and may exist alone or in combination within any given device. For example, the interface allows a computer to be a talker, a listener and a controller. A voltmeter might be only a talker, only able to supply data to the system, while a printer may be only a listener, only able to accept data from the system. Additionally, these functions may be active or not at any given time.

Figure 1 illustrates how an interfaced system might work. The lines on the right of the figure represent the signal lines of the bus. There are a total of sixteen signal lines, divided into three groups. The first group is composed of eight data lines, forming the data bus. These lines are bidirectional and carry information and messages between devices.

The data byte transfer control group is composed of three lines called DAV (dava valid), NRFD (not ready for data), and NDAC (not data accepted). As the name of each of these lines implies, this group is used to control the flow of information over the data lines. The five remaining lines form the general interface management group. These lines are used for control and status information pertaining to the devices attached to the bus.

JUST BOUGHT A TRS-80?

By now, those who took advantage of Tandy's reduced price on the Model 1 computer have discovered two things. One, the computer is excellent value, two, audio cassette program storage is not!

So we think you'll be ready to consider Stringy Floppy. Storing and retrieving programs at an amazing 700 characters per second entirely under computer control. No controls to operate, no volume level to set!

Then after we've told you about Stringy Floppy, we'll explain how our Internal Memory can triple available Random Access Memory (RAM) for well under $200 installed. And finally, for those to whom this sounds like salvation, a money-saving package deal offer.

Stringy Floppy is a compact high speed tape transport system that plugs into the 40 way expansion connector on the back of your keyboard. A 2 for 1 adaptor is included to allow other equipment, such as a printer, to be plugged in at the same time. Using the TRS-80's SYSTEM command, the SF becomes part of the TRS-80, adding its own instructions to those of BASIC.

SF records digitally onto miniature cartridges called "Wafers", about the size of a credit card and 5mm thick. Wafers contain an endless loop of special tape and store from 4,500 bytes (characters) to 64,000 bytes, depending on length. All lengths are $3.50 each except the longest (23 metres) which are $4.50 including Sales Tax.

The SF Drive is mechanically robust for long life. Just slide in a Wafer and it "clicks" home. Capstan and head assembly are fixed in a die-cast aluminium drive, and the "endless loop" means no rewind capability is required. Simplicity means reliability.

SF's Operating System allows you to certify Wafers and load or save up to 99 BASIC or machine language files on a Wafer. Up to 8 SF Drives may be connected to a TRS-80. The Data I/O Program included adds commands to store and retrieve data rather than programs. Optional programs allow saving of programs/arrays/screen memory complete (@FREEZE $10) or emulate a full disk operating system with named files/passwords/directories/random records (BASIC OPERATING SYSTEM $27). Access at 25 cms (10 inches) per second Wafer speed, and 700 cps transfer rate is FAST. With SF programs can chain under computer control. A variety of useful software exploiting these capabilities is available from ASP. We even include with SF a Monitor program specially designed to help you transfer machine language cassettes to SF Wafer.

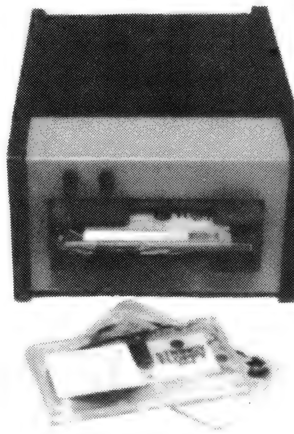
SF is supplied with Data I/O and Monitor programs, a Wafer Holder, 10 mixed length Wafers, a 2 for 1 Bus Extender, 6 month limited warranty, and our Stringy Floppy Newsletter. Price is $371.00 including Sales Tax, add $6 for freight within Australia.

And now back to INTERNAL MEMORY, an amazing professional quality plug-in module which fits INSIDE the TRS-80 keyboard and expands memory to a very useful 48K. If you have one of the new Japanese made TRS-80s (see label underneath) INTERNAL MEMORY is $186 installed. For the earlier USA made units it is $152 plus $10 fitting if required. In both cases we include a super machine language memory test on Wafer so you can thoroughly test your computer and minimize servicing worries. Again our 6 month warranty applies and $6 delivery within Australia.

PACKAGE DEAL

As promised, temptation time. Buy both STRINGY FLOPPY and INTERNAL MEMORY together and you pay $499 to suit USA keyboards, or $523 to suit Japanese units. Included in both cases is installation of your INTERNAL MEMORY, our popular Data Base Manager (FMS) normally $20, and freight back to you. Either way you save $60! Just let us know by phone that your keyboard is coming, and send it freight prepaid.

Bankcard purchases are welcome. You must quote full Card Number, Expiry Date, and sign your order.



IBM TYPEWRITERS

IBM builds excellent typewriters. From their well known Selectrics Models 1, 11, and 111 to their latest fully electronic Models 50, 60 and 75. They have superb keyboard feel, are robustly constructed and hold their value. In fact their only obvious shortcoming is that they won't connect directly to a computer.

ASP solves that shortcoming for ALL the above models.

The Selectric models are electro-mechanical in operation, so ASP provides a precision mechanical assembly which fits inside the typewriter to actuate it internally. Detailed installation/adjustment instructions are included for those who wish to do their own installation. Alternatively ASP offers a $100 fitting service with fast turnaround.

Models 50 and 60 are fully electronic, so installation to the typewriter is more straightforward. Basically the typewriter is opened up and special connectors plugged into the IBM electonics. Again detailed instructions are included, or ASP's $50 fitting service is available.

The same electronic assembly, in different versions, controls all typewriter conversions. It is packaged in a compact box, about the size of a house brick. Controls for power on/off, hold/run/self test are incorporated on the front panel. The assembly, known as ASP's Universal Controller, is microprocessor based and accepts data on either the RS232 Serial Standard, or TTL Parallel basis. In RS232 mode, baud rates of 110, 150, 300, 600, 1200, 2400, 4800, and 9600 are available. Handshaking may be RTS, RTS inverted or XON/XOFF. In Parallel mode handshaking may be Busy and/or Data Accepted. A data buffer of 96 characters is provided.

All conversions are "print only" devices, operating at 13.5 characters per second in the case of Selectrics, and 17.5 cps in the case of Models 50, 60 and 75. The special capabilities of IBM's fully electronic models are also available to the host computer, as though the instructions had been typed on the typewriter keyboard.

And pricing? Including Sales Tax, to suit Selectric models - $740 and to suit the Electronic Models 50 and 60 - $499.

Use as a typewriter when not required by the computer is definately not affected, and there are no external alterations required in any conversion.

Freight within Australia is $6, and you may order on Bankcard by sending us a signed Order quoting your full Card Number and Expiry Date.

ASP MICROCOMPUTERS

797 DANDENONG ROAD, EAST MALVERN 3145.
VICTORIA AUSTRALIA.

Telephone: (03) 211 8855 211 8344

Legal Corner: - TRS-80 and Selectric are doubtless Trademarks of their respective Manufacturers, Tandy and IBM.

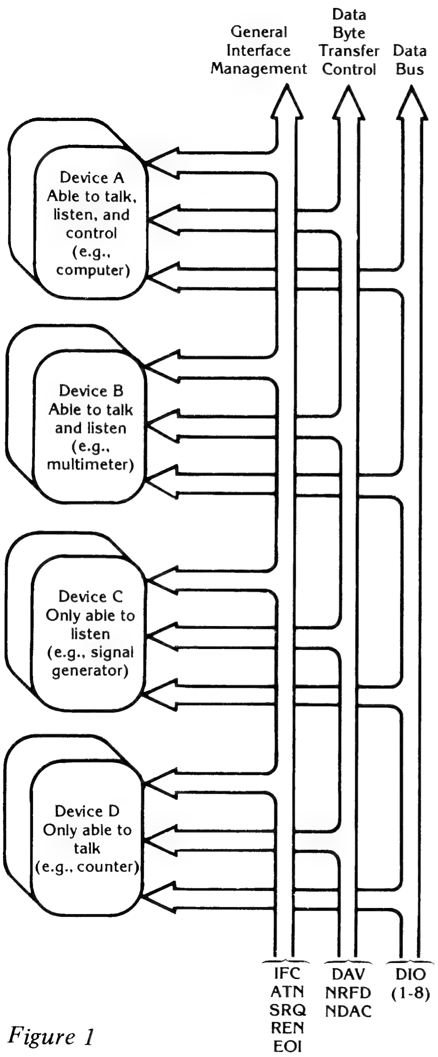


Figure 1

ASSIGNING ROLES

Four devices are shown attached to the bus in figure 1. Device A is a controller, a talker and a listener. As a controller, it may assign the role of the active talker to any device on the bus capable of being a talker, including itself. As a talker, it can supply information to other instruments on the bus, and, as a listener, it can receive information from a talker.

Although device A is shown as the only controller in this example, more can be accommodated in a system. Only one controller can be active at a time, however, to prevent conflicts. A controller that is designated system controller becomes active at power-up. All others must remain passive until control is passed to them. The IEEE standard specifies the signals and timings necessary to do this.

Device B is both a talker and a listener. It can be addressed by the controller and made an active talker. The active talker has control of the DAV control line in the data byte transfer control group. Device C can only be a listener, and can be addressed by the controller and made an active listener. Device D is only a talker and can be made an active talker by the controller.

Active listeners have control of the NRFD and NDAC lines in the data byte transfer group. The active talker drives the data lines and the active listeners read the information. To avoid conflicts, only one talker is allowed to be active at a time, but several listeners may be active simultaneously.

TRANSFERRING INFORMATION

The potential existence of several active listeners receiving data simultaneously presents a problem. The active listeners may not be capable of accepting data at the same rates. The speed of information transfer must be controlled by the slowest active listener in order for data not to be lost. The data rate sequencing is controlled by an electronic voting system called "open collector". The transfer of information takes place as follows:

1. All active listeners indicate on the NRFD line their state of readiness to accept a new piece of information. The line is usually connected to a voltage through a resistor. This resistor causes the line to assume the same voltage on both sides of the resistor if there are no loads on the line. If an active listener is not ready, it turns on a transistor connected between the NRFD line and ground. The turned-on transistor acts like a short, pulling the voltage of the NRFD to ground. The active talker will not transmit the next data byte until the voltage on the NRFD line reaches its high voltage condition, when all of the active listeners have become ready and have released the NFRD control line.
2. The active talker, having put valid data on the data lines at least two microseconds (0.000002 seconds) ago, asserts the DAV line by pulling it low. Two microseconds is a settling time to allow the data to reach valid logic voltage levels. Assertion of the DAV line is a signal for the active listener(s) to read the information on the data bus.
3. During the previous portions of the data transfer, the NDAC line has been held in the low state by the active listeners. When DAV is asserted and the active listeners accept the data, they will release their hold on the NDAC line. When the last active listener releases its hold, the line will be pulled high by a resistor.
4. The active talker waits until it observes the NDAC line in the high state, signifying that all active listeners have accepted the byte. It then releases the DAV line to end transfer. The release of the DAV line is the cue for the active listeners to again pull down on the NDAC line in preparation for the next data transfer.

A timing diagram of the complete handshake is shown in figure 2. Note that control of data transfer is effected by the active talker and active listener(s). Once the controller has configured the bus, it takes no part in the data transfer.

CONFIGURING IT OUT

Now that we have examined how data is transferred on the HP-IB, let us consider the operation of configuration. One of the general interface management lines is called ATN (attention). This line, run by the active controller, signifies whether the information on the data lines is for control or data transfer.

When the controller wishes to configure the HP-IB, it asserts the ATN line. This causes any active talker to relinquish the DAV line. The transmission of control information is similar to that for data, but the active controller takes the place of the active talker and both talkers and listeners accept the information.

The active talker and active listeners may be designated during the transmission of this control information. The information is on the data lines. The following table shows what these values are:

Bit number	7 6 5 4 3 2 1 0
bus command	X 0 0 C C C C C
listen address	X 0 1 L L L L L
talk address	X 1 0 T T T T T
secondary address	X 1 1 S S S S S

Note that bit 7 is not used (DIO 8). Bits 6 and 5 are used to designate one of four classes of control information. A bus command (bit 5 = 0, bit 6 = 0) is used to directly control the devices on the bus. Such functions as triggering of devices and passing control require bus commands. Listen addresses (bit 5 = 1, bit 6 = 0) are used to activate listeners. A listener that observes its listen address on the bus when attention has been asserted becomes an active listener. The state of other listeners remains unchanged.

UNLISTENING

Thirty-one listen addresses are possible, from 0100000 to 0111110. The last code in the listen address class, 0111111, is the unlisten command. All active listeners become inactive when an unlisten is sent. Talk addresses (bit 5 = 0, bit 6 = 1), are similar to listen addresses except that the definition of an active talk address causes any other active talker to become inactive, since only one active talker at a time is allowed.

The 1011111 pattern is the untalk command, leaving no active talkers on the bus. Secondary addresses (bit 5 = 1, bit 6 = 1) are used to address subunits within a device. Some devices may provide more than one simultaneous function and require more extensive addressing than the talk and listen addresses provide.

The remaining four lines in the general interface management group are used to control the interface sections of the devices. IFC (interface clear), is used by the active controller to override all bus activity and put the bus into a known state. Such an action

is abortive to any data transfers in progress and is used when something has gone wrong.

REN (remote enable) is a line that allows the bus to control a device. The active controller indicates whether an addressed listener will use programming information sent to it by a talker by using REN.

EOI (end or identify) is used in two ways. It may be asserted by the active talker to designate the last byte in a data transmission, and it is used during a parallel poll, discussed later. SRQ (service request) is a line which a device may use to get the attention of the active controller. Note that this is a request, not a demand, and may be ignored by the active controller until there is time to service the request.

When the controller decides to acknowledge the service request, it has to discover which device on the bus issued the request. Since all devices on the bus share the SRQ line, all service requests look alike.

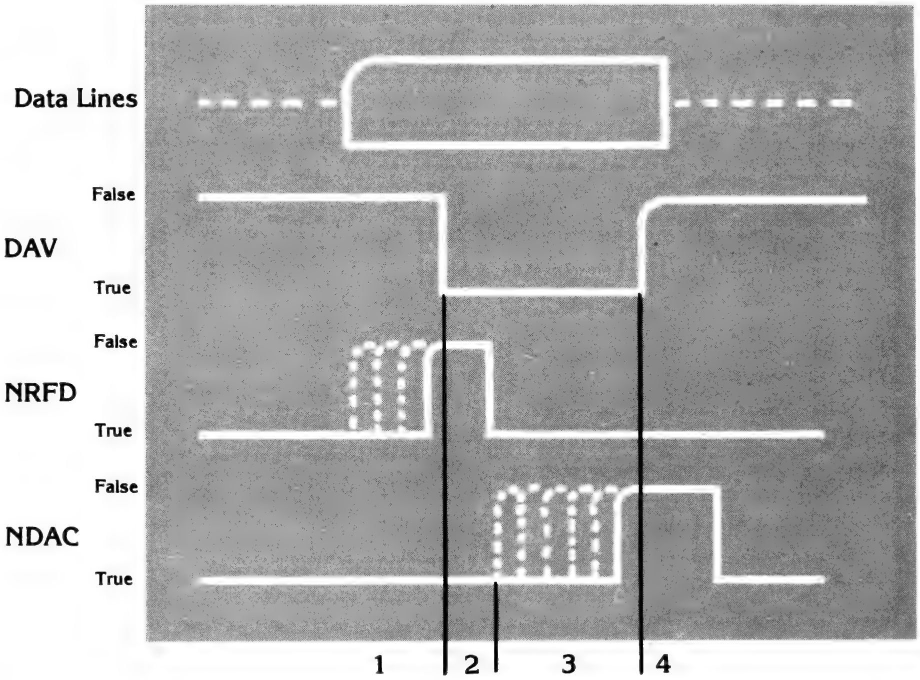


Figure 2

POLLING ALONG

There are two ways the active controller can determine the address of the requesting device. Both methods are polls. A poll is a request for status information. The active controller may request the status of a device individually, by addressing the device as a talker, and sending the device a serial poll enable command, one of the bus commands sent while ATN is asserted. The active controller can then obtain eight bits of status information about that device. Serial poll disable must then be sent to return the device to the data mode.

For faster decisions, a parallel poll may be made. The active controller asserts ATN and EOI, thus requesting a parallel poll. Up to eight devices may respond, each one using a different data line (DIO1 to DIO8). If a device is requesting service, it will pull down on its data line, signifying that condition.

YOU DON'T HAVE TO KNOW

One of the best features of the IEEE standard is that a system user need not know any of the preceeding information. It is built into the definition of the interface and is supposed to work correctly for any device built to the IEEE specifications. What, then, does the system user need to know?

The actual messages and data formats sent are not specified. They are service dependent. For example, a voltmeter wishes to inform the desktop computer that it is reading +1.433 volts at its input. When it becomes the active talker, what should it send? Since most computers recognize the ASCII character set, it would be nice to send ASCII. The only decisions left are the format and the order of the digits, least-significant to most-significant, or the opposite.

Most computers prefer the most-to-least-significant order, and the voltmeter would send +,1,.,4,3,3,CR,LF. The CR and LF characters stand for carriage return and line feed, two characters used to terminate a transmission.

The definition of messages and message formats leaves the area of the IEEE standard and enters the realm of specific busses, which is the HP implementation of the standard. And that removes yet another level of interfacing problems from the shoulders of the system user.

BACK ISSUES SERVICE

Here is a complete guide to available back issues of APC. A quick guide to their contents is shown here. Check the coupon for the issues you require.

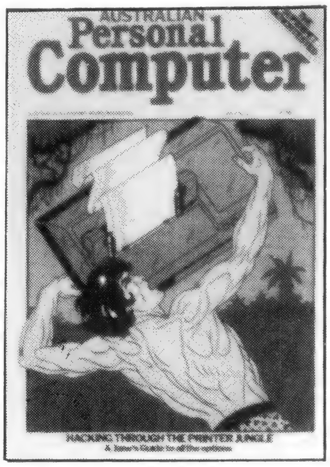
Volume 1 No. 3
Benchtest: Tandy TRS-80 Model II. Animistics. M68000-Motorola's Sweet 16. Computer Games: Big trees and the Alpha-Beta algorithm. Plotting in Three Dimensions. The Complete Pascal: Loops. Checkout: Exatron Stringy Floppy. Z80's control signals. Programming the Simple Approach. Z80 Instruction Set. Programs: Byte Saver (TRS-80), TRS-80 Graphics, Word Processor (Sorcerer), Space Slalom (Apple), Robot Nim (PET).

Volume 1 No. 5
Limited supplies only available
Benchtest; Sinclair ZX80. Marketing Strategies of Micro Manufacturers. Computer Games. Gateways to Logic: Boolean Algebra. Micro Chess: A brief history of chess programming. The Complete Pascal: Control Structures II. Whistling up Colour: Apple "hands off" control. Programs: Showpiece (Apple), Dots & Boxes (PET).

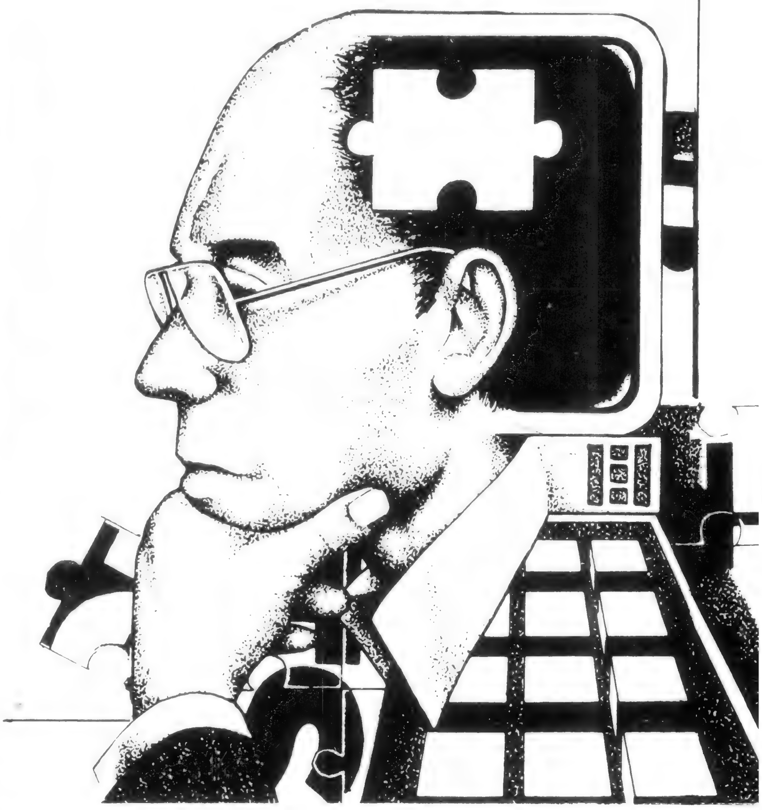
Volume 1 No. 7
Robotics. Micro Chess. Computer Games: Guessing the odds. APC-80. The Complete Pascal: Records and files. Calculator Corner: Speed up your TRS-80 pocket computer programs. Programs: TRS-80 Tarot, PET Bloobers, PEEK & POKE for Apple Pascal, PET Demolition, Superboard Bug Bypass, String Function, Several Sorts.

Volume 1 No. 8
Benchtest: Atari 400 & 800. The Complete Pascal. Gateways to Logic. APC-80. Helping the Handicapped: A look at a US company specialising in this field. Programs: Reading 'System' Tapes into the System 80, Monster Multiplier (Apple II), Read-write Routines (PET), Program Formatter.

Volume 1, No. 4
Benchtest: Hewlett Packard's HP85. Benchtest: Texas TI 99/4 Home Computer. Gateways to Logic. Adding a different family of chip to your micro. Computer Games: Speeding up the Search. The Complete Pascal: Data Structures. Programs: 3D Noughts & Crosses (TRS-80), PET Backgammon.



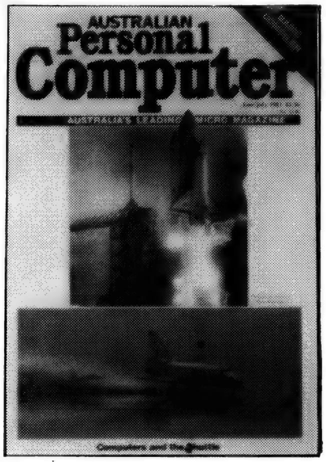
Volume 1 No. 6
Benchtest: CBM Model 8032 'SuperPET'. Microchess. Basic Basic. Benchtest: SuperBrain. Programs: Lunar Lander (TRS-80), PET Fighter Pilot, Apple Plotting, Lprint to Print Utility (TRS-80), ZX80 Breakout, Graph (TRS-TRS-80).



Volume 1 No. 9
The Last One: The program which writes programs. Multi-user Systems. Computer Games. Disk Basics: What makes them click? Get Well Soon: How to recover from a data tape disaster. APC-80. Gateways to Logic. Ultra-fast Tape Storage: Data-transfer rates of up to 9600 baud for the Ohio Scientific Superboard. The Complete Pascal. Forth: A review of the language. Programs: TRS-80 Target Practice, TRS-80 Four in a Row, PET Anagram, PET Obstacle Course, Minefield (TRS-80).

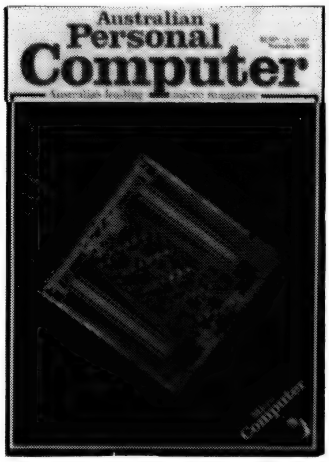
Volume 1 No. 10
Slow Scan TV: How micros are putting themselves in the picture. Disk Basics: More on disk drive interfacing. APC-80. On The Level: Cure those tape troubles. Computer Games: Draughts. The Complete Pascal. Choosing a Computer System: Hints on buying your first business computer. Word Processing: How to define your needs. Profile: Rudi Hoess, the man behind Apple in Australia. Multi-user Systems. Systems Development: Program development and implememtation. Searching String Arrays. Turning USR(X) Routines into Data Statements. Pack It In: Squeeze more data onto your disks or tapes. Programs: TRS-80 Rocket Attacks, Superboard Mines, PET Brick Stop.

Volume 1 No. 11
Benchtest: Acorn Atom. Switch in Time: For when 'Dallas' is too important to miss . . . Gateways to Logic. Another Dimension: Three-dimensional plotting. The Complete Pascal. APC-80. Computer Games: Chess. Wordprocessor Benchmarks. WP Benchtest: Word Pro 4 Plus. Multiuser Test: MP/M. Systems Development: Profile: Dick Smith. Program Packing. Single Key Keyword Entry: Superboard. Search/Change in Applesoft. Put Your Hooks into OSI Basic. Programs: Sorcerer Simple Word Processor, Compucolor II Function Plotting, TRS-80 Auto Fuel Consumption, TRS-80 Towers of Hanoi, PET Aircraft Landing, TRS-80 Giant Trap, PET Wire, PET Convoy.

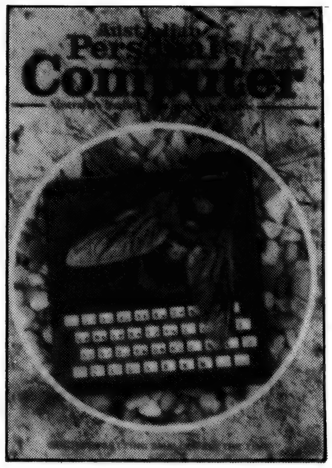


Volume 1 No. 12
Inside Printers. Computer Games. Benchtest: Bigboard, Gateways to Logic. TRS-80 Compiler. OSI Basic-in-ROM. WP Benchtest: Magic Wand. Profile: Nigel Shepherd. Parliament House in Micro: The role of Cromemco. Multi-user Test: MVT/Famos. Solar System Simulation. Drawing Lines on PET's 80 x 50 Grid. Single Key Key Development: Abbreviating commands. Programs: PET Replace, TRS-80 Demon Hunts, PET Chords, ZX80 Sliding Letters.

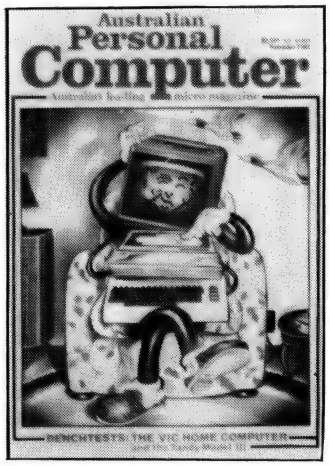
Volume 2 No. 1
Printer Review: Dick Smith's offering. Gateways to Logic. Joystick Interfacing: Control on the TRS-80. Classroom Zest: A computer aided instruction package described. TRS-80 Compiler. APC-80. The End of Flowcharting. Wordprocessor Comparison: Spellbinder vs Wordstar. Multiuser Test: Onyx C8002. Choosing a Programmer. Profile: The Tandy Corporation. Unix: A Review. Define Your Own Function Key on PET. Superboard II Edit: Extra function key. Multiplexing PET: Increasing I/O data handling. Programs: Program Packing (6502), Cosmic Wars '80 (TRS-80), TRS-80 Worm, System 80 Dodgem.



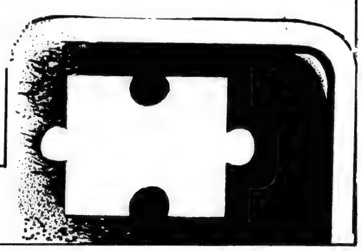
Volume 2 No. 2
Apples for the Handicapped. TRS-80 Tiny Compiler. Profile: Bill Anderson. Facing the Future: Barry Jones. Print Using for Applesoft. SBII Plus: Improve the Superboard II. Multiple File Tape Backups. Relocating OSI Basic-in-ROM. Programs: Bigboard Real Time Clock, APC-80 Alien Invasion, PET Radio Technician Course.



Volume 2 No. 3
Benchtests: Sinclair ZX81. APC-80. Building a Bigboard. Keystroke Reduction: TRS-80. WP: The Sanders Printer. Profile: Bill and Niki Saunders. CP/M by Rodnay Zaks. Rapid Bubble Sort. Encryption with RND and USR. The 6502 Dream Machine. Vectors and the Challenger IP. Programs: TRS-80 Input Routine, Treasure Hunt (PET).

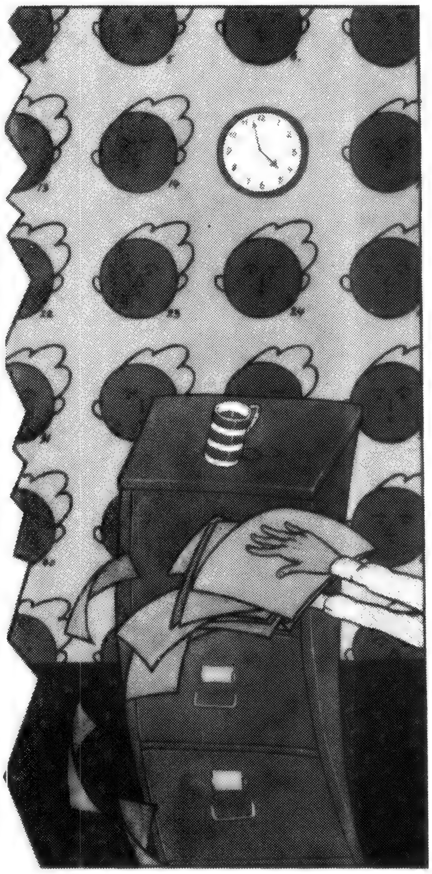


Volume 2 No. 4
Benchtest: Vic 20. Software Review: MCMON. Computer Games. Tree Access Routines. Gateways to Logic. Is it a Bird? What is a personal computer. How Computers Communicate. Benchtest: Tandy III. Profile: Gary Blom. Program Planning. 6502 Assembler in Basic. Increase Your Word Power. Programs: PET Arithmetic Test. Apple Mondrian.



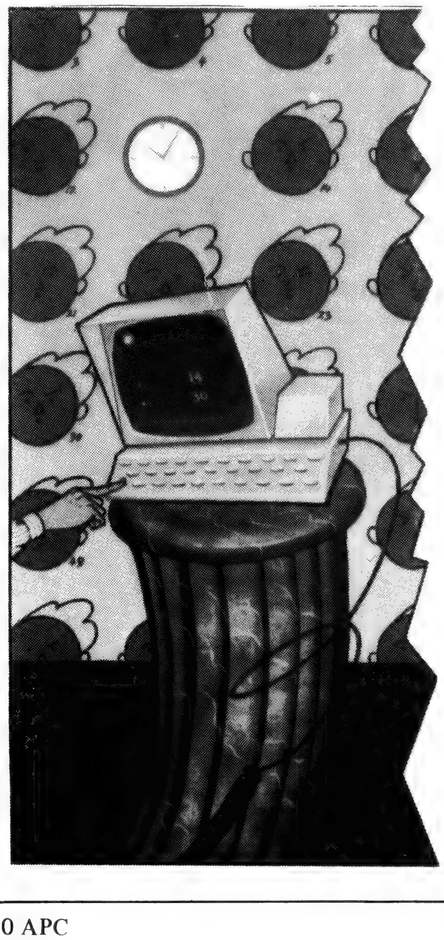
MICROCOMPUTER DATABASES

What is a database and how does it differ from an ordinary file handling system? Lyn Antill explains the principles involved.



A database, in principle, is simply a collection of data — my diary, for instance, is a database of my activities over the year. In principle, the data need not be stored in any given fashion; in practice, however, database material is stored on a computer in a particular way, with the advantage that the data can be interrogated in a variety of ways. One need not, for instance, start reading from the beginning to the end to find the answer to a specific question; on the contrary, a special suite of programs called a Database Management System (DBMS) will enable such a task to be performed.

You can ascertain whether a DBMS will suit your requirements by looking hard at the sort of data you wish to store and the way in which you want to be able to get at the information. This is known as data analysis, and is the first task carried out in designing a large computer installation. I therefore propose to begin with a couple of examples of data processing programs in order to give a feel for what is involved.



Firstly, I want to look at payroll programs. During the course of a week, or a month, all amendments to payroll data — rises, overtime, etc — are collected on file. This is sorted into similar order to the main file and, as each payslip is produced, a quick look at the update file will show whether there are any amendments to be made before processing.

There is a very good reason why this was one of the first applications to be computerised in many organisations, which has nothing to do with whether it would save time or money, or make the business more efficient. It is simply that the first method of computer data storage was to use magnetic tapes with the records stored in sequence. This is precisely what you want when you are producing one payslip after another for each of your employees.

A different record-keeping problem is posed by a stock control system in a warehouse. At any moment, you may want to know whether any one of your lines is in stock, but there may be no time to start your search at the beginning of a file. Thus, a direct access method was introduced with the advent of the disk. If you know the part number, you can arrange the physical location of the records so that the part number corresponds to the position, or you can keep an index of which records are stored where, which will enable you to read the required record directly.

But what if you don't know the part number? Perhaps the question that needs to be answered is: 'How many different types of nail do we stock?' If you were regularly having to answer questions like that you would want a quicker way of getting the answer than trawling through the whole file in the hope of getting the few records that you needed. This is where DBMS will enable you to keep track of your data records from more than one point of view. The more questions your database system can answer for you, the more efficiently you can run your business and the more flexible you can be in these difficult times.

Before launching into a discussion of how a DBMS works, it is probably worth bringing the non-programming reader up to date on the way in which conventional file handling is carried out by some of the more common programming languages and operating systems.

Background

The Basic language treats file storage as though it is just one long list of data items which are read one at a time into program variables. Although a Read instruction might permit several of these to be collected at once, they still have to be named individually. They also have to be named individually when they are written on to tape or disk in the first place. This makes for program statements which are longer than necessary, and it also demands of the programmer that he always bear in mind every field on the file, even if he is only interested in a few of them.

Pascal, Cobol and many enhanced Basics have 'records' which are logical groupings of data items. These are read or written as a single instruction. Cobol has particularly sophisticated record-handling features because it was designed specifically for record processing.

Records can be arranged in different ways within a file. In a serial file the records are written one after the other as they occur, and they are written nose-to-tail on the tape. A sequential file is also nose-to-tail, but the records are in some logical sequence, eg, alphabetical order; the sequence is determined by reference to some 'key' field. The CP/M operating system also supports 'relative' files. These are not arranged nose-to-tail but are at specific locations on the disk, so that they can

COMPAK
COMPUTER SHOP

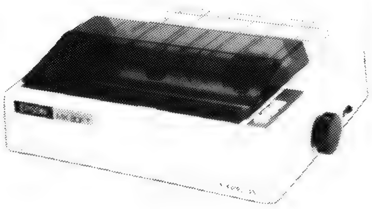
BRIGHTON
592 6285 TEL.

44 THE ESPLANADE
BRIGHTON BEACH
VIC. 3186

DANDENONG
793 4555 TEL.

81A FOSTER ST.
DANDENONG
VIC. 3175

TWO LOCATIONS



EPSON PRINTERS

MX-80	$949 incl. tax
MX-80 F/T, 2	$1095 " "
MX-100	$1395 " "

ALL CABLES & I/FACES EX STOCK

DISKS

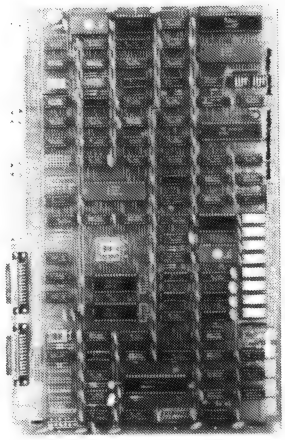
8" Slimlines	$635
5" MPI & TANDON	
40 Track SSDD	$300
40 Track DSDD	$390
80 Track DSDD	$510
Single Case W. Supply	$85
Double Case W. Supply	$110

DISK PRICES & TAX

NOW COMPAK HAS TWO SINGLE BOARD COMPUTERS FULLY ASSEMBLED & TESTED

- Z80A, 4MHZ, 64K
- 8" and/or 5" DSDD Floppies
- On-board Video
- On-board Keybd CCTS
- 2 RS-232 Ports
- Centronics PTR. Port
- QRNS CP/M 2-2

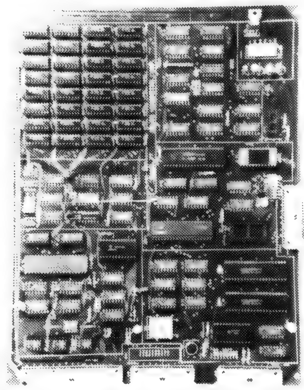
Model V — 10



$990 & Tax

- Z80, 2MHZ, 64K
- 8" DSSD Floppies
- PROM Writer/PAR. Port
- Centronics PTR. Port
- Kybd & Video via RS-232 Port to Terminal
- Runs CP/M 2-2

Model V — 711



$795 & Tax

OEM's
PLEASE
NOTE!

TYPEWRITERS AS PRINTERS

COMPAK ARE LEADERS IN THIS FIELD AND INTERFACE:

OLIVETTI	PRAXIS 35
	ET-121
	ET-221
ADLER	1010
	1030
OLYMPIA	ES100
IBM GOLFBALL	

CALL FOR INFO/PRICES

DEALERS & END USERS

PLACE YOUR ORDERS NOW FOR:

1. APPLE DISK DRIVE AND CONT. FROM JAPAN ONLY $500 & Tax RET.
2. KEYBOARD IN CASE WITH 10 FUNCTION KEYS AND ENCODING. ONLY $180 & Tax

(Also connects to Apple)

WE MAKE A **TRS-80*** **MODEL 1 DISK CONTROLLER** TO RUN OFF EXPANSION PORT.

PRICE only $195 incl. tax

WE CAN ALSO SUPPLY DISKS AND POWER SUPPLY TO SUIT.

INTRODUCTORY OFFER

COMPAK

TRS-80 is a Radio Shack trademark.

APPLE is a trade-mark of APPLE inc.

ARCHIVES AUTHORISED DEALER

HAZELTINE ESPRIT TERMINAL
$890 & Tax

TRS-80 MOD III DSDD DISKS
$3350 incl. Tax

DIABLO DAISYWHEEL
ONLY
$2950 incl. Tax

MICROCOMPUTER DATABASES

be located directly. This means that the program can pick out, say, record number 23.

Cobol often also includes indexed files, in which the records are stored more or less in sequence on a key field, but with gaps left to slot in later additions, and each has an associated index file. This has a list of the keys and their respective addresses in the main file and, being much smaller than the datafile, it can be searched far more quickly.

Sequential (and serial) files are the easiest to program, the most compact, and the quickest to read right through. Relative files (which are sometimes erroneously called random files) are still fairly easy to program, provided your language and operating system support them. They do rely on your knowing the position of each record you want to access, although it is possible to start at record 1 and search through if you have to. Indexed files involve an operating system overhead in that the system has to search through the index first before locating the record. Every time you insert a new record or delete an old one, the index has to be updated. As with relative files, it is still possible to start at record 1 and go right through. Indexed files are not difficult to program provided your language has them — indeed, they are the most common arrangement on large computers. They are better than relative files in that they permit the use of naturally occurring keys such as account numbers or part codes which saves you altering your manual procedures to fit in with the computer.

Disadvantages of conventional file handling

If you are using a sequential file, all processing must be performed in the order in which the records are stored, so you might have to read through the whole file in order to answer a question about one record. You will also have to rewrite the whole file if you change a single field. With the payroll example, we actually wanted to plough through each record in turn and create a new copy of the file with the updated figures. We deliberately kept changes and overtime payments on a serial file of their own — adding new items onto the tail of the file as they cropped up, and then sorting the whole 'change file' into the same order as the payroll file. This saves having to make new copies of the payroll file other than on pay day. If we have to answer the occasional query like 'How much does Mr Fraser earn?', then we resign ourselves to spending five minutes reading through from Mr Adams.

Relative and indexed files both presuppose that you know the key of the record you're looking for so that if a file's keyed on account number and you only know the person's name then you're back to searching the whole file again. If this only happens occasionally it might not matter, but in some situations you are continually asking questions about your data where it would obviously be nonsense to expect to know the key of a particular record in advance.

An example of this is provided by our student records system. The following questions are asked regularly:
— Has John Smith paid his fees?
— Has anybody dropped out of Computer Studies?
— Is anybody in room 463 on Wednesday afternoon?
— How many married women are there on engineering courses?

Each of these requires a different searching technique. The first requires that we locate a particular record. The second requires that we locate all records of students registered on Computer Studies and seeing whether any of them carried the code which indicated that they had withdrawn. We can only find whether room 463 is occupied if our database carries records about rooms and timetables as well as students — it would obviously be useful if it could. To find out about married women engineering students, we have to search on three keys — sex, marital status and course. Which order we do these in will depend on which order the records are stored in. Since students are often recognised by the course they're on, we'd probably trawl through all the engineering courses totting up all the records where sex = F, and marital status = M.

We have 10,000 students and their data could all be fitted on to a micro with a hard disk, though it's obvious that when we have a queue of students and teachers wanting information from the system, we would have insufficient time to read through the file from the start every time someone asked a question.

What is a database?

The word 'database' covers a range of ways of holding data on a computer. It differs from a conventional file in that you can retrieve any piece of data in more than one way. You are not obliged to know beforehand the unique key number of a record or to start at the beginning and keep going until you find it.

It takes a bit of tricky programming to keep track of data in a complex database, so only in extreme circumstances would a programmer bother to write his own routines. You can buy a Database Management System which will cope with most of your needs, although nobody would be rash enough to say that his system would cope perfectly with everything — if it did, it would either be too large or too expensive, or probably both!

The very simplest sort of DBMS is an extension to your language and operating system, permitting you to keep more than one index to an individual file. I haven't seen a micro system like this yet, but many minis offer ten or more different indices. It won't be long before many micros offer two or three. This would be enough for many micro-sized information handling problems. A really sophisticated DBMS will permit you to store complicated interrelationships of different types of records. It will have its own data dictionary to let

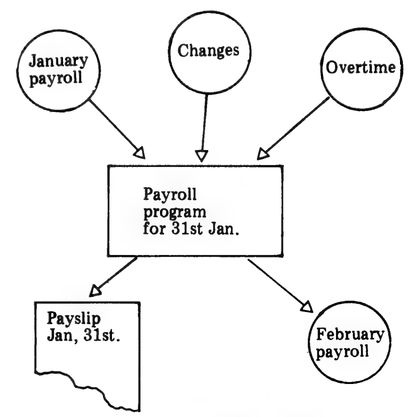


Fig 1 Run chart for payroll program. This is a traditional batch process using sequential files.

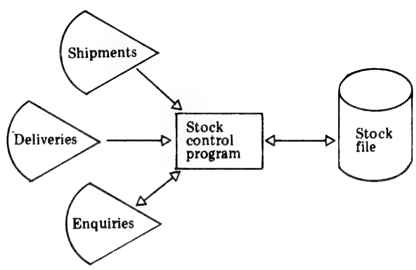


Fig 2 Run chart for stock control. This uses random access disk files which can be accessed by several different programs at the same time.

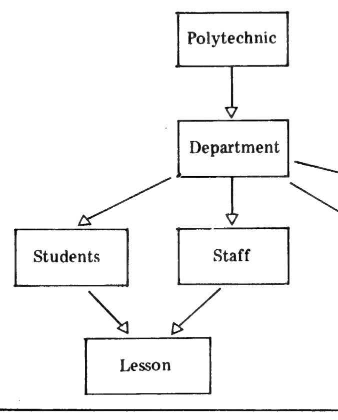


Fig 3 Type Diagram' for the Polytechnic Database. This shows the different types of records stored and the pointers. There are four sets of pointers from each Department record and two to each lesson record.

MICROCOMPUTER DATABASES

you define the data you want to store and its own 'query language' to enable questions to be typed in at the keyboard and answers displayed without the need to write any programs at all.

There are several ways of arranging the data so that it can be retrieved in a variety of ways. Experts argue about the exact categories but there are three common types — multi-indexed (also known as inverted), hierarchical, relational. All sorts of terms are thrown about by the manufacturers, and it is often difficult to tell what exactly you're being offered.

Multi-indexed

This is the simplest to implement and, on a micro, is probably the most satisfactory for general use. One of the fields in the record is nominated as the primary key, and the records are stored in sequence on that key. This key might be a relative address or it may be the key to a primary index (as with indexed sequential files). Other fields may then be nominated as secondary keys. An index is kept for each of these keys. Primary keys often have to be unique, secondary keys do not. The most serious limitation of this system is that all your records must be of the same type, ie, they must all have the same fields in the same places. On my student database it would prevent me answering the question about which students were in room 463, because I could have student records but not timetable records.

A popular example of this type of database interrogates you to find out what data formats, keys, etc, you want in the first place, then it gives you a menu of possible actions – insert and delete records, search on keys, sort on keys. In fact, it has three types of key. The primary key is purely arbitary and refers to the record's physical location within the file. Each record may then have up to about five 'key fields'. Records can be retrieved on these fields and the file can also be sorted on them. Suitable candidates for key fields would be Name, Date of Birth. Finally there are "attribute fields'. You can list a large number of possible attributes (nearly 200 on my version) of which an individual record may have as many as ten. The system keeps a list of which records have which attributes. Finally, each record has room for several lines of text which could be anything you wanted to include in the record.

The demonstration disk that came with the program held a personnel file. The primary key was the employee number, the sort keys were surname, first name, date of birth, department, and the attribute fields included things like relevant skills (plumber, carpenter, French, oil-wells). If a vacancy came up for an electrician to work on an oil-well in Algeria, you would first ask the system to find all the electricians, select from those all who had worked on oil-wells, then all those who spoke French or Arabic. Hopefully, it would then be able to give you a list of suitable people. Only at this point would you start looking at complete records. You could also ask it who had the highest salary, because salary is a sorted field.

This sort of database is also known as inverted and the process of listing which records have a certain value in a certain field is known as inversion.

Hierarchical

The same standards people who wrote Cobol came up with the Codasyl extensions to Cobol to define a standard database. There have been attempts to put this onto a microcomputer database. There is, somewhat to my surprise, a Codasyl database for CP/M. It is called MDBS. It is expensive and slow on floppy disks, but it is worth knowing about because it points up the limitation of inverted files.

The major advantage is that a variety of records can be kept. On the polytechnic example, we could have records on departments, students, staff, classrooms, courses. The logic of the system goes as follows: The poly owns a set of departments, each department owns a set of courses and each course owns a set of students. Each department also owns a set of staff. But there are more relationships than this. A staff member owns a set of students, ie, all the students he teaches. This will be some or all of the students on one or more courses.

A set is essentially part of a file. The owner of a set and its members are linked together by means of pointers, ie, alongside the poly record is the address of the first department record (Accountancy), next to this record is the address of the next department record (Architecture) and so on until the last department record which points back to the poly record (just so you know you've got to the end and haven't lost anything).

You've already discovered the first disadvantage of this method — it's complicated. If you're setting up a large database, anything you do is going to be complicated, but you probably won't want to be bothered on a micro, unless it is important to you to be able to link records of several different types. If you do see a package which permits you to define several different types of records with a 'this owns that which owns the other' relationship between them, then it could well be a Codasyl database.

Relational

The relational database is a theoretical construction. The underlying work is heavily mathematical -- indeed a whole new branch of mathematics (the relational calculus) was invented to describe it. Although it works amazingly in principle, it has proved difficult to implement in practice. The intention is to be able to relate any data item to any other data item, with none of the artificial limitations that are imposed by the files and records of traditional data processing. I should just be able to pour all my data into the computer without any presuppositions about relationships. This means that I have to get rid of all the assumptions that have already been made, consciously or not, in my existing system. This leads to the systems analyst's biggest hassle — getting the data into fourth normal form, which means that all the fields in a record are dependent on 'the key, the whole key and nothing but the key.' and, moreover, that there are no significant relationships between the subordinate data items except their dependence on the key. For instance, Name and Address are firmly related in most people's minds, but if you are ever to be able to change someone's address then they must *not* be related in the database.

I don't propose to launch into a full description here — there are plenty of textbooks on database design that cover the subject in the sort of detail that is required to understand it. The important point for the potential user is that the relational database — if it truly deserves that name — is not something that can be set up by an amateur. Even a small system requires a professional data analysis — that's even harder to come by than a professional programming job.

There are, however, several microcomputer databases that call themselves relational. I very much doubt if the purist would accept them as such. This hardly matters, because they will stand or fall by what they enable the user to do with his data. They are only likely to be useful to the non-professional if they are used for data which is already 'normalised', ie, where you want to keep a lot of straightforward records with a minimum of analysis required to get them into different types. The ones that I have seen are not obviously very different from inverted file systems.

Conclusion

Successful implementation of a database system depends on a clear understanding of the records that are to be kept, and the questions that are to be asked about them. You may find, particularly with an inverted file, that you can only keep one type of record. If you have several types of records with complex relationships between them, you need a hierarchical system or a network system which is even more complex.

For the user, there is also a major decision to be made, namely whether the system is to be used as a stand alone — ie, an electronic filing cabinet — with its own data definitions and query procedures, or whether it is to be accessed by programs for further processing of the data. The type of database doesn't really matter here because all three types could be interfaced to programs or to their own query languages, or indeed to both. Programming is generally managed by means of calls to external subroutines to perform database functions. Each call includes a parameter list which includes such things as the keys which are being searched on, and the storage location into which the answer is to be put.

One difficulty for the prospective purchaser is that few of the software salesmen have experience of databases, which have been the preserve of big machines until now. This means that you are quite likely to find someone who can't answer technical questions. However, it is probably just as effective to stick to the pragmatic questions: 'What sort of records can I store?'; 'How can I access those records?' END

NEW! Apple software from Micro-Educational

APPLE SOFTWARE APPLE PERIPHERALS FROM MICRO–EDUCATIONAL

Australia's leading mail-order supplier for Apple Software and Peripherals. We offer wide range, phone advice, fast delivery ex stock, 10-day money-back warranty on Software and very competitive pricing.

HARDWARE (Prices exclude sales tax. Add 17.5% S.T. if applicable.)

ANDROMEDA 16K EXPANSION CARD	195.00
MICROSOFT Z80 SOFTCARD	379.50
PAL COLOUR CARD	195.00
CCS ASYNCH SERIAL CARD	185.00
CCS COMMUNICATIONS CARD	210.00
CCS PARALLEL CARD	159.50
CCS A/D	145.50
CCS CLOCK/CALENDAR	149.50
CCS COMMUNICATIONS CARD	185.00
CCS GPIB IEEE 488	320.00
CCS ARITHMETIC PROCESSOR	495.00
MH CPS MULTIFUNCTION CARD	275.00
WESTSIDE SUPERCLOCK	195.00
ELECTROMED SENDATA COUPLER	325.00
PAYMAR LOWER CASE	49.95
16K RAM UPGRADE	49.95
BLANK DISKS (10 IN LIBRARY BOX)	45.95
LIBRARY BOX	4.50
T&G PADDLES	44.95
JOYSTICK	69.95
ABT BARCODE READER	199.50

SOFTWARE (Tax included. Sales Tax exemption does not apply.)

ADMINISTRATION

PFS	109.50
PFS: REPORT	109.50
VISICALC 3.3	219.50
VISIPLOT/TREND	295.00
VISIDEX	219.50
VISITERM	175.00
VISIFILE	275.00
DESKTOP PLAN II	219.50
TARGET BUSINESS PLANNER	249.95
REAL ESTATE ANALYSER	175.00
CONTEXT CONNECTOR	195.00
THE GENERAL MANAGER	109.95
WINCHENDON STATISTICS	220.00
DB MASTER	245.00
DB UTILITY PAK #1	109.95
WORDSTAR	349.50
SPELLSTAR	165.00
SANDY'S WORD PROCESSOR	175.00
SPELLBINDER	495.00
THE WORD	99.50
SUPERSCRIBE II	139.95
THE DICTIONARY	109.95
SCHOOL TIMETABLE	49.95
TIME MANAGEMENT	195.00
MICRO MEMO	39.95
CORP SYSTEM	190.00
DBASE II	700.00

EDUCATION

APPLE PILOT	175.00
FRACTIONS	39.95
DECIMALS	39.95
ARITHMETIC SKILLS	49.95
COMPU–SPELL	39.95
ELEMENTARY MATHEMATICS VOL 1	49.95
CARTELS & CUTTHROATS	43.95
SENTENCE DIAGRAMMING	24.95
THE LINGUIST	39.95

SIMULATIONS AND ADVENTURES

FLIGHT SIMULATOR	34.95
THREE MILE ISLAND	41.95
WARP FACTOR	43.95
OPERATION APOCALYPSE	64.95
SHATTERED ALLIANCE	64.95
BATTLE OF SHILOH	64.95
CONFLICT	27.50
SOUTHERN COMMAND	43.95
OLYMPIC DECATHLON	29.95
GLOBAL WAR	29.95
HIRES 0 (MISSION ASTEROID)	23.95
HIRES 1 (MYSTERY HOUSE)	27.50
HIRES 2 (WIZARD & PRINCESS)	37.50
HIRES 3 (CRANSTON MANOR)	41.95
HIRES 4 (ULYSSES)	39.95
HIRES 5 (TIME ZONE)	109.95
SOFTPORN ADVENTURE	32.95
ULTIMA	43.95
ZORK	37.50
ZORK II	37.50
OO TOPOS	37.50
CYBORG	37.50
GALACTIC EMPIRE	29.95
GALACTIC TRADER	29.95
GALACTIC REVOLUTION	29.95
TAWALA'S LAST REDOUBT	32.95
CRUSH CRUMBLE CHOMP	32.95
TEMPLE OF APSHAI	29.95
AKALABETH	39.95
STAR WARRIOR	42.95
ROBOT WAR	39.95
CASTLE WOLFENSTEIN	32.95
DUNGEON CAMPAIGN/ WILDERNESS	34.95
DRAGON'S EYE	24.95
DARK FOREST	32.95
STONE OF SISYPHUS	32.95
TARTURIANS	29.95
GOBLINS	29.95
CREATURE VENTURE	29.95
WIZARDRY	49.95
ALKEMSTONE	43.95
KAVES OF KARKHAN	43.95
SORCEROR OF SIRA	31.95
ODYSSEY	29.95
DUNGEON CAMPAIGN/ WILDERNESS	34.95
SARGON II	37.95
BACKGAMMON	31.95
BRIDGE	29.95
MONOPOLY	44.95
HI–RES GOLF	32.95

ACTION AND ARCADE GAMES

RASTER BLASTER	32.95
POOL 1.5	37.50
GORGON	42.95
APPLEOIDS	32.95
PHANTOMS FIVE	37.50
SPACE EGGS	32.95
STAR THIEF	32.95
FALCONS	32.95
APPLE PANIC	32.95
SNOGGLE	29.95
HEAD ON	27.50
INTERNATIONAL GRAND PRIX	32.95
AUTOBAHN	32.95
FENDER BENDER	29.95
HIRES SOCCER	34.95
MISSILE DEFENSE	32.95
BILL BUDGE SPACE ALBUM	44.95
EPOCH	37.50
SNEAKERS	32.50
PULSAR	32.95
STAR CRUISER	29.95
ORBITRON	32.95
OUTPOST	32.95
CYBER STRIKE	43.95
COPTS & ROBBERS	32.95
SPACE QUARKS	32.95
SPACE RAIDERS	32.95
ABM	29.95
GENETIC DRIFT	32.95
TRACK ATTACK	32.95
BUG ATTACK	32.95
FIREBIRD	32.95
DAVID'S MIDNIGHT MAGIC	37.50
RED ALERT	29.95
SHADOW HAWK	34.95
BEER RUN	29.95
RINGS OF SATURN	43.95
ESCAPE FROM ARCTURUS	39.95
SHUFFLEBOARD	32.95
GAMMA GOBLING	32.95
PEGASUS II	32.95
HADRON	32.95
THIEF	32.95
THRESHOLD	43.95
CROSSFIRE	32.95
MAZEMAN	32.95

GRAPHICS

BUDGE GRAPHICS	43.95
SUBLOGIC 3D GRAPHICS	69.95
A2–3D1 ENHANCEMENTS	29.95
A2–3D2 GRAPHICS EDITOR	39.95
APPLE WORLD	69.95
HIGHER TEXT	39.95
HIGHER GRAPHICS II	39.95
COMPLETE GRAPHICS SYSTEM	69.95
DATASOFT MICROPAINTER	39.95
EZ DRAW 3.3	57.50
PASCAL GRAPHICS EDITOR	109.95
ARCADE MACHINE	49.95

UTILITIES & LANGUAGES

GRAFPAK	95.00
EXPEDITER COMPILER	109.50
HAYDEN COMPILER	195.00
TASC (MICROSOFT COMPILER)	195.00
TRANSFORTH II	135.00
ALD SYSTEM II	135.00
APPLE PILOT	175.00
APPLE FORTRAN	220.00
MICROSOFT FORTRAN	220.00
FORTH II	99.50
SUPERKRAM	179.50
PHIL'S CHEAP EDITOR	59.95
TINY PASCAL	54.95
LISA ASSEMBLER 2.0	62.50
LISP	109.95
MONITOR EXTENDER	27.50
DISK FIXER	32.95
DOSOURCE 3.3	49.95
TELLSTAR	85.00
APPLE–CILLIN	49.95
ACTION SOUNDS & HIRES SCROLLING	24.95
ASCII EXPRESS	79.50
Z–TERM	169.95
THE SOURCE	140.00
VISITERM	175.00

BOOKS

ASSEMBLY LANGUAGE	19.95
BENEATH APPLE DOS	24.95
APPLE II USERS GUIDE	19.95
APPLE MACHINE LANGUAGE	17.50
MOSTLY BASIC	17.50
PASCAL PROGRAMMING (LEWIS)	17.50
APPLESOFT LANGUAGE	12.95
APPLE BASIC FOR BUSINESS	19.95
COMPUTER GRAPHICS PRIMER	19.95
SHAKE HANDS WITH THE APPLE	12.50

SUBSCRIPTIONS

INFOWORLD (WEEKLY AIRMAIL)	120.

SUBSCRIPTIONS

INFOWORLD (WEEKLY AIRMAIL)	120.00

For our free catalogue send the attached coupon or write to:
MICRO–EDUCATIONAL, 17 PARK ROAD, GARDEN SUBURB, NSW 2288.

MICROLINE 80 PRINTER

DUE TO AN ENORMOUS DEMAND WE HAVE EXTENDED OUR SALE PERIOD.
WE APOLOGISE TO ANYONE WHO MISSED OUT LAST MONTH.

ONLY $565 Tax inc. ex stock.

- 80 CPS
- 9 x 7 Dot Matrix
- Pin Feed and Friction Feed Standard
- 3 Character Sizes
- Centronics Interface Standard
- Graphics
- Upper and Lower Case

* SUPERBOARD II ADAPTOR/ CABLE . . . .$30 Tax inc.

* TRACTOR FEED OPTION . . . .$110 Tax inc.

* SERIAL INTERFACE . . . .$190 Tax inc.

BOX OF PAPER . . . .$34 Tax inc.
(2000 sheets, 9½", 11")

RIBBONS TO SUIT M/LINE 80 . . .$2 each

SUPERBOARD II . . . . . $395 & Tax
(includes DABUG III & 8K RAM standard, prepaid membership to KAOS, the largest 6502 user group in Australia)

B/W T.V. MONITOR. . . . . . . $125Tax inc.

MPI B51 DISK DRIVE . . . . . . $295 & Tax

MPI B52 DISK DRIVE . . . . . . $395 & Tax

QUME DATATRAK 8 DISK DRIVE . . . . $635 & Tax

DISCOUNT FOR QUANTITY

bankcard welcome here

WHAT'S NEW AT COMP-SOFT?

THE PEACH COMPUTER

Great value for money. **$1495.00 + TAX**
The computer designed for everybody — from the student to the business user.

FEATURES THE MOST EXTENSIVE BASIC ON ANY MICRO THAT WE HAVE SEEN.
SUPER HIGH RESOLUTION IN COLOUR (6040 x 200)

COMES STANDARD WITH:—

- Complete RS 232 Interface
- 80 or 40 Column Display
- Parallel Centronics Interface
- Light Pen Input
- 32 K Ram (expansion to 64 K)
- 24K Rom Basic
- Internal 6 Slot Motherboard
- Numeric Keyboard
- Sound Generator
- User Definable Keys
- 6809 CPU
- R.G.B. and B/WComposite Video Output

ETC. ETC. ETC. WOW!!!

IF YOU DON'T THINK THAT'S WORTH THE MONEY, YOU'RE CRAZY!!

COMP-SOFT MICROCOMPUTER SERVICES
235 SWAN STREET, RICHMOND, VICTORIA, 3121. TELEPHONE (03) 428-5269

The most important feature of a small computer...

...is where you buy it.

It's the key to making the most of your microcomputer system. Because the right computer store will provide you with the support, service, software and supplemental equipment that can make the difference between total satisfaction and total frustration with your system.

Filling this vital role was the concept behind ComputerLand. Our purpose is to provide you with ready access to the entire world of microcomputing. Today we represent over 50 different manufacturers and suppliers. But no matter what brand or type of equipment you purchase or lease from Computer-Land, you get an extra feature no one else can offer — the resources of the world's largest network of computer stores.

Information. At ComputerLand, you deal with a staff of thoroughly trained and experienced professionals. We know small computers. And since we deal only in small computers, we know them a lot better than anyone who also has to know CB radios, burglar alarms, stereos, TV sets, copy machines and typewriters.

We can help you choose the computer system that best meets your needs. If you have an installed system, we can tell you which accessories and programs are compatible.

We can also put you in touch with computer clubs, classes, and other customers who have situations similar to yours.

Choice. ComputerLand has the widest selection of small computer products you'll find anywhere. And every product we offer has been evaluated and approved by our expert product selection committee.

We have no ties to any single manufacturer, so we don't have to try to sell you any single brand. We can combine computers, accessories, and programs from several different sources to meet your specific needs. You don't have to adapt your needs to fit our system. At ComputerLand, you have complete freedom of choice. Because we do. And as your computer needs expand, we have the add-ons and programs that can make your computer expand as well.

Support. The right computer store becomes even more important after you buy. Because that's when you'll need warranty service, routine maintenance, technical assistance, training, software — or just someone friendly to call when you have a question. ComputerLand provides all of the above and then some.

We give your equipment a complete checkout before it leaves the store. We can do most repairs in-store so you don't have to wait while your equipment makes a round trip to the manufacturer. And if the equipment is needed for business, we can provide priority service. So come to ComputerLand and let us introduce you to small computing. We want to be your computer store.

ComputerLand®

Brisbane	— 1st Floor, 1 Adelaide Street, Phone 211-97777
Canberra	— 22 Colbee Court, Phone 822-342
Gold Coast	— 126 Scarborough Street, Phone (07) 221-9777
Melbourne	— 123 Lonsdale Street, Phone 662-2133
South Melbourne	— Cnr Albert Road & Kingsway, Phone 266-647
Brisbane	— 1st Floor, 1 Adelaide Street, Phone 211-9777
Perth	— 5 Mill Street (Across from the Hilton) Phone 321-4671
Sydney (Bondi)	— 119 Oxford Street, Phone 389-4466
Sydney (Chatswood)	— Chatswood Plaza Shop 3, Phone 411-7611
Sydney (City Centre)	— 31 Market Street, Phone 290-2955
Sydney (Parramatta)	— Victoria Road at Church Street, Phone 683-3199
Sydney (Ryde)	— Top Ryde Shop Centre, Phone 808-2666
Sydney (Burwood)	— 31 Burwood Road, Phone 745-4311

For other locations opening soon Ring (02) 27-8959

NEWSBRIEF

by Graeme Domm.

ATTACK ON GOVERNMENT

The Software Industry Committee of the Australian Computer Society has presented a range of recommendations to the Industries Assistance Commission seeking, amongst other things, Federal Government action in matters such as provision of venture capital, tax incentives, and the formation of a high technology advisory group for Australia.

The committee, one of several formed by the Australian Computer Society, presented its submission last month, with the full support of The Association of Australian High Technology Industries.

AAT Association spokesman Robert Kopp said he agreed with the Software Industry Committee when it referred to the "failure of the government to support the local EDP industry".

The submission made particular reference to two current government projects: " the awarding of the contract for the Australian bicentennial network to a foreign supplier; and, secondly, in Victoria, a recent tender for software for the department of Minerals & Energy (which) was not published".

Mr. Kopp said Governments had paid "lip service" to their local preference purchasing policies.

The Software Industry Committee submission points out that none of its recommendations were for protection from overseas software companies, nor, for artificial support for the industry. "Rather, the recommendations call for a viable commercial and social infrastructure to allow the software industry to take its full role in the economic and social well-being of Australia until the end of this century and beyond", the submission concluded.

COMPANIES JOIN FORCES

Impact Computers and Onyx Computers (Aust.) have established a joint venture partnership to market Onyx microcomputers throughout Australia.

The new company, Onyx Australia Pty. Ltd., has been granted rights from Onyx Systems Inc. of California, U.S.A. to act as authorised distributor of the Onyx range of microcomputers in Australia.

Douglas Broyles, President of Onyx Systems Inc., U.S.A., in confirming the appointment, said: "Onyx systems is pleased to have an organisation of the quality of Onyx Australia Pty. Ltd. to represent and support our products in Australia. We feel that this is a significant market for our products. Rapid developments occurring within the Australian computer industry have been of great interest to us. We are confident that the wide experience and skills of Onyx Australia personnel will keep ahead of these changes and will also ensure the continuing success of Onyx products in Australia."

Impact Computers and Onyx Computers have consolidated their marketing, sales and technical facilities under the Onyx Australia banner. National Sales and distribution outside New South Wales will be controlled from the Onyx offices in Dee Why, Sydney. Impact in Parramatta will be responsible for New South Wales sales and distribution.

Personnel movements within the new organisation include the appointment of Mr. Grahame Wood as Managing Director of Onyx Australia.

One of the benefits to emerge from the association, according to the two companies, is the establishment of a national hardware maintenance support service through STC.

To coincide with the joint venture Onyx Australia will be launching a national media campaign promoting the Onyx C5001, C8001/MU/8002 microcomputers and the new desk top SUNDANCE model. The company has a range of standard and customised applications programs.

Onyx is aiming its sales thrust at the smaller business enterprise or educational facility which has outgrown its first computer and needs a multi-user system without going to, or in some cases being able to afford, a larger main-frame system.

DESKTOP WITH W.P. PRINTER

The 800 XR, a desktop business computer offering data processing, communications and word processing in a single system, has been released in Australia by Durango Systems.

The new computer adds word processing capabilities to Durango's 800 series of business computers. A feature of the 800 XR is its high resolution printer and powerful Star Text word processing software. In its data processing mode, the 800 XR produces economical computer printouts and document drafts at up to 200 characters per second. Then at the touch of a switch it can print letters on letterhead and envelopes, Durango says.

The printer is engineered for high throughput. The carriage performs with high-speed tabbing, minimising its movement through bidirectional "Look-Ahead" capabilities and uses a life-long print head. Because of the special paper handling mechanism, the printer has the capability of printing anywhere on the surface of the page and can completely blacken a 20cm by 30cm sheet of paper.

The printer provides a choice of font styles.

For everyday drafts, listings, and reports, an operator can use a standard or compressed type produced with a single pass of the printhead using a nylon ribbon. The ribbon comes in an easily replaced cartridge and usually lasts months under normal usage. The printer is equipped with a built-in continuous form-feed mechanism with adjustable width. An adjustable thickness control accommodates up to six-part forms.

The Durango Star Text software package provides tools necessary for efficient and timely document generation, modification and printing. Star Text has a document library facility and can be linked with Durango's SBA accounting system file for automatic selective letter output.

Star Text also offers graphic capabilities. It can be used to generate graphs, pie charts and other visuals.

The 800 XR consists of an Intel 8085 8-bit microprocessor with 2K bytes of ROM, 65K bytes of RAM, and dual floppy disk storage up to 2M bytes. The system also offers a 1920 character CRT, a keyboard with a 10-key numeric pad, and the dual-mode bidirectional matrix printer.

The 800XR also incorporates a second microcomputer and two intelligent controllers to control and improve the performance of the printer, diskette drives and display screen.

Also part of the basic package is the DX-85M real time operating system, Star Basic compiler/interpreter, utilities and diagnostics. CP/M operating system is also available on certain models.

Durango Systems is at 21 Chapel St., Marrickville, N.S.W. (02) 517 1300.

END

CISA IS PROUD TO BE SYDNEY'S FIRST DOWNTOWN HITACHI DEALER

PEACH PERSONAL COMPUTER FEATURES

USER MEMORY
32k – 1MB RAM

PERMANENT MEMORY
24k ROM

MICROPROCESSOR
High Speed 8/16 bit HD6809

ANCILLARY STORAGE
5¼" 80k, 5¼" 640k (dual), 8" 1MB, 8" 2MB (dual), 5, 10, 20, 40MB Hard Disk (Winchester)

KEYBOARD
86 Keys from which 262 symbols and functions are produced.
10 Function Keys (5 with shift)
Tactile Feedback
23 Key Numeric/Directional Pad
All keys are easy to read and use

SCREEN
Super high resolution 640 x 200 dots
40/80 characters x 25 lines
Upper and lower case
Green phosphor and many other monitors available

DIAGNOSTICS
Power-on self testing
Parity checking
Complete Diagnostics kit available

COLOUR GRAPHICS (STANDARD)
8 colours
262 characters and symbols
Simultaneous graphics and text
640 x 200 dots, 320 x 200 dots, 160 x 100 dots, 80 x 100 dots

LANGUAGES
MICROSOFT COLOUR BASIC (standard)
Machine Language (standard)
CIS COBOL
CP/M (with use of Z80A card)
PASCAL USDC
PASCAL ISO
059 Basic
FLEX
FORTRAN
Over 150 games, 30 graphics packages, word processing, hicalc, integrated financials all AUSTRALIAN produced

COMMUNICATIONS
RS232
CENTRONICS
Light pen
Colour video
Monochrome video
Six expansion slots
up to 9600 baud
OPTIONAL
IEE 488
Analog to Digital
Digital to Analog

PRINTERS
80 characters/second
180 characters/second
Letter quality and more . . .

MISCELLANEOUS STANDARD FEATURES
Clock
Speaker with volume control
Full documentation
Games/Graphic diskette/cassette
Membership to
THE AUSTRALIAN SOURCE
Membership in PEACH USER'S GROUP
Vast range of peripherals supported
UNEQUALLED HITACHI QUALITY
90 day warranty followed by full board level replacement by any dealer
Interactive software – computing made simple

CISA
CALL OR VISIT NOW:
89 YORK STREET, SYDNEY.
Ph: (02) 29 1599

Communication

Address letters to 'Communications', Australian Personal Computer, P.O. Box 115, Carlton, Vic. 3053.

SHARGON PATCH

I have experimented with a patch program for SARGON 11 Chess, running on a TRS80 or System 80 enabling this program to save and load unfinished games, board positions and chess problems, etc. I have also developed a data file on cassette containing 50 authentic chess problems. For further details, send SAE to B. Fripp, 30 Castlecor St, Ferny Grove. 4066.

COMPUCOLOR IN SA

I am a Compucolor II owner and an avid reader of the Compucolor Column in Australian Personal Computer, keep up the good work.

Your column has highlighted a few truths i.e. the user of a home computer cannot possibly hope to utilise the full capacity of his machine on his own. Therefore I think it's about time a Compucolor user's group was established in Adelaide. I'm sure there are quite a few owners hidden away who would love to share their programs and knowledge with others.

Therefore I would like to ask, if you would include a brief note, asking any Compucolor owners in South Australia who wish to form a users group to contact me at P.O. Box 63, Ingle Farm, S.A. 5098. or telephone 260 3693 after hours.

Thank you for your time with this matter and I hope we may contribute something to Compucolor owners in Australia.

T. Wardle

SLEEP LEARNING

Your magazine A.P.C. is great reading and thank you for the marvellous job you are doing.

Having just finished reading your "Letters" department, I wondered if either you or anv of your many readers might be able to help me solve a small problem. The problem I speak of is a peripheral that could be attached to a Sytem 80 and used to control a tape recorder, enabling sleep learning. Although I would prefer a single device without having to use a computer, I realise these are far too expensive and limited.

And since I own a System 80 (I have already tried Dick Smith's, no luck), the program presents no problem. I hope that some one can assist me with some information on this subject.

Thank you for any help that you can give me and keep those mag's coming.

Patrick Hilton, West Footscray, Vic.

LONG LIVE COMMODORE

Despite the recent demise of the Commodore Victorian Group, there is definitely a great need for a Commodore Group to form in Victoria - especially after the VIC 200 arrives. So if there is anyone out there who would like to organise such an association, please do so. Now that the Commodore is well known and used here and Commodore in Sydney are willing to assist User Groups, there should be a large following.

Nicki Saunders

APC-80 PROBLEMS

I am fascinated by the uses of the 'Move' command and the various functions it can perform. Is there some way of using this to save an entire screen onto tape?

I have typed in all the mods for the Basic listing of A.P.C.-80. When I first typed in Version 1 I found that if any error (Besides the APC-80 commands) occured my micro would 'freeze up' (I have a TRS-80 model 1 level 11 16K micro). I had thought this was one of the drawbacks associated with the advantages of extra commands, until recently I found that this should not occur. I have concluded that I must have either made a typing error or missed one of the issues. Could you please - in the next edition of A.P.C. – publish the full, updated Basic listing - indicating where the various routines start (especially the TRAPPR routine).

Mladen Bank.

Your suspicions were indeed correct, APC-80 should not cause the machine to lock up when an error occurs.

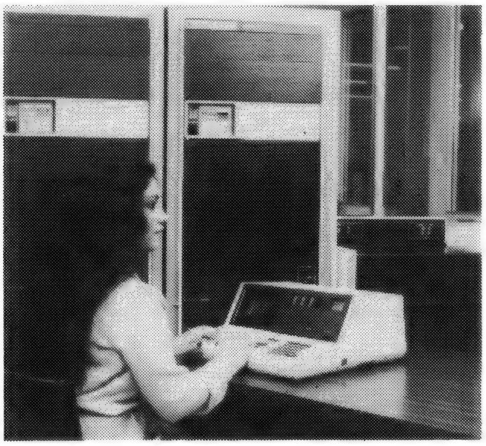
By a happy co-incidence, this issue of Australian Personal Computer contains an article called "APC-80 Rides Again", which includes a full listing of the program, both Basic and assembler.

Concerning saving screens to tape, your best approach would be to use the APC-80 system tape generator and supply parameters which specify the video RAM. This data file could then be re-loaded using the SYSTEM command.

LET THE

AUSTRALIAN SOURCE

BRING MAINFRAME POWER INTO YOUR HOME OR OFFICE



WE ARE PROUD TO ANNOUNCE THE LAUNCHING OF AUSTRALIA'S FIRST MICROCOMPUTER INFORMATION UTILITY

By popular demand!!!
Special offer held over until
29th of February

The AUSTRALIAN SOURCE is Australia's first microcomputer information utility, aimed at giving the average microcomputer user access to the computer data banks and also massive storage space previously only available to large Mainframe installations. The era of home computers has now officially begun in Australia, now that the AUSTRALIAN SOURCE is here; as now, any small businessman or student can have on his desk for an extremely small cost, a system that has the level of storage power and access to data banks for which only a few years ago, government departments and large private users were paying millions of dollars.

The AUSTRALIAN SOURCE can be accessed through a number of 'approved' personal computers, and terminals, by use of an accoustic coupler or modem.

Users will receive a number of benefits which will include:

INFORMATION SERVICES

Members will have access to a number of information sources which will include the latest news, sports results, financial reports, and farm information.

NATIONWIDE ELECTRONIC MAIL

Members will be able, also, to communicate nationwide with other AUSTRALIAN SOURCE members through our electronic mail system.

THE AUSTRALIAN SOURCE RECOMMENDS 3M DISKETTES

SOFTWARE BANK

Users will have access to a myriad of computer programs that will include entertainment, educational aids, programming and diagnostic tools, and financial applications.

MAINFRAME POWER

Users will have the capability to make use of the Mainframe's huge storage capacity by using any of our large programs, or storing your large programs on our system.

SHOP AT HOME

You can take advantage of our 'shopping by computer' system to get the best prices on a number of popular consumer items.

EXTREMELY EASY TO USE

You do not have to be a computer programmer to make use of the AUSTRALIAN SOURCE. All of the instructions are in everyday English, so that even the younger members of your family will be able to operate the system.

LOW COST

While services like these used to cost tens of thousands of dollars to the government departments and large corporations who used them, they are now available to you for less than the cost of a packet of cigarettes a day for the 'average' user.

You can join the AUSTRALIAN SOURCE by paying a one–time joining fee (normally $100) and a small hourly user charge (normally $10 an hour 8 a.m. – 6 p.m. and $4.50 an hour 6 p.m. – 8 a.m.).

SPECIAL CHARTER MEMBER OFFER

As a special promotion, the joining fee before the 1st February, 1982, is only $60 and this also entitles you to 20% off all list user charges in the future – a great deal for the money.

TO JOIN THE AUSTRALIAN SOURCE FAMILY, AND TAKE ADVANTAGE OF THE EXCITING SERVICES WE WILL BE OFFERING, PLEASE COMPLETE THE COUPON BELOW:

MEMBERSHIP REQUEST

Yes, I think the idea of the AUSTRALIAN SOURCE is great! I enclose $60, so please send me my user manual and password.

☐ ***I have a computer, Brand........................***

Model........................

☐ ***Please send me information on low cost equipment packages I can use to take advantage of the AUSTRALIAN SOURCE'S services.***

NAME: ....

ADDRESS:...

CITY:...

STATE:................. POSTCODE...............

PHONE....................

Mail to

THE AUSTRALIAN SOURCE PTY. LTD.,
364 LaTrobe Street,
Melbourne, Victoria, 3000.
Tel: (03)329.7998

☐ ***I am interested, but I need more information.***

DEALER INQUIRIES INVITED

MORE IN EVERY... XEROX 820

Not Just Another Leading Brand Computer!

On first glance our Xerox - 820 Information Processors appear no different to any other leading brand business computer: It is made by a well known reliable company, it has inverse video, business forms displays and is accompanied by a library of reliable well-supported application programs. Our Xerox 820 however, is demonstrably ahead of other leading brand business computers.

Cost Efficient!

We would not want you to buy it because it is the cheapest, but the price v.s. performance aspect must be considered. When you are doing that you should also take into account our FIVE YEAR WARRANTY option! That is not all; our system will grow with your business. We can supply a Xerox 820 System with one terminal and an 8" Dual Diskette drive and expand same to a multi-terminal 110 M.byte system without any obsolensence! You get the best from day one at the right price. Check the others!

The Novex CD–2000 Operating System.

Whilst we are sure that you will find the system very competitively priced both on Hardware and Software there are other angles you want to compare us on: Apart from being able to run programs which are supported by CP/M, our Xerox 820 also runs the Novex CD–2000 Operating System! This is great news for businessmen who like getting the most out of their investment. The CD–2000 Operating System is the most flexible, most advanced system currently available for Micro Processor based systems. To begin with, it allows the running of up to five programs simultaneously! It is like having five computers in one! Experience shows that operator machine efficiency can be increased by as much as 300% due to CD–2000 exclusive features. Add the fact that your operator could get more out of your system, but requires less training, you now know you have a winner!

The Novex CD–4000 Data Management System.

Another exclusive feature! The CD-4000 Intelligent Data Management System is not just another 22 M.byte Winchester Technology Disc. The system is Micro-Processor Controlled and is capable of executing within itself certain utility programs – like running its own sorts without utilising the host processor! The system creates and moves files on command, flags bad sectors if any and generally runs its own housekeeping! Up to 8 "Host" systems may be interfaced to the CD–4000 all sharing the same Data Base. Even larger systems may be configured, sharing an even larger Data Base. Parallel and Serial ports are available as options as well as an IEEE port. The System manages Sequential, Index Sequential and Regular files of varying record and field lengths.

Nationwide Support and Distribution.

Novex Data Systems (Aust) Pty. Ltd. is the Appointed Distributor of the Xerox 820 in Australia. Through its National Dealer Network of 35 Regional Deaiers and 9 Specialist vertical market Dealers, Novex Data Systems (Aust) Pty. Ltd. offers excellent Support and Service in YOUR Area. Ring us now to obtain the phone number of a Dealer near you !

NOVEX
MORE IN EVERY BYTE

NOVEX DATA SYSTEMS PTY LTD 418 ST KILDA ROAD MELBOURNE VICTORIA AUSTRALIA 3004 TELEPHONE (61.03) 267 5544

Melbourne	Sydney	Perth	Adelaide	Wellington	Auckland
03–267 5544	02–27 4974	09–322 2338	08–212 7737	72 6382	77 5059